Cadence 使用参考手册

邓海飞微电子学研究所设计室

2000 年 7 月

目录

概述

作为流行的 EDA 工具之一，Cadence 一直以来都受到了广大 EDA 工程师的青睐。然而 Cadence 的使用之繁琐，又给广大初学者带来了不少麻烦。作为一位过来人，本人对此深有体会。本着为初学者抛砖引玉的目的，本人特意编写了这本小册子，将自己数年来使用 Cadence 的经验加以总结，但愿会对各位同行有所帮助。本册子的本意在于为初学者指路，故不会对个别工具进行很详细的介绍，只是对初学者可能经常使用的一些工具加以粗略的介绍。其中可能还请各位同行加以指正。

1.1 Cadence 概述

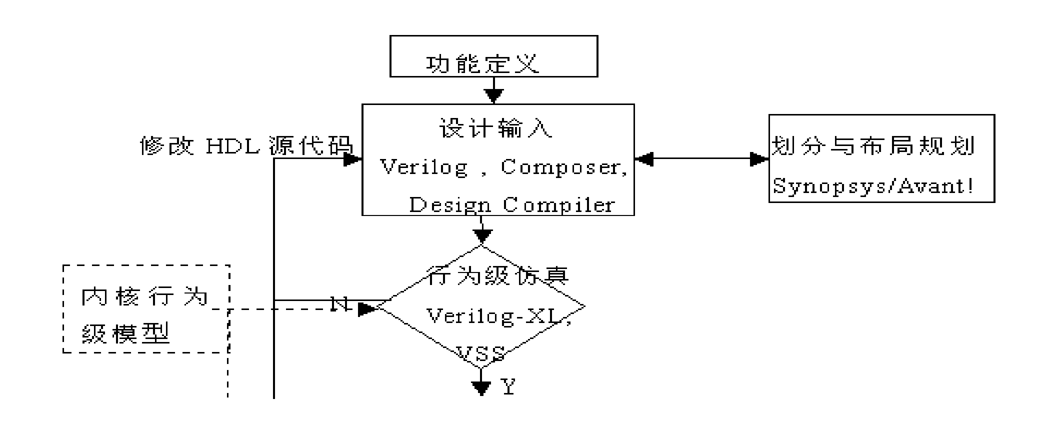
Cadence 是一个大型的 EDA 软件，它几乎可以完成电子设计的方方面面，包括 ASIC 设计、FPGA 设计和 PCB 板设计。与众所周知的 EDA 软件 Synopsys 相比，Cadence 的综合工具略为逊色。然而，Cadence 在仿真、电路图设计、自动布局布线、版图设计及验证等方面却有着绝对的优势。Cadence 与 Synopsys 的结合可以说是 EDA 设计领域的黄金搭档。此外，Cadence 公司还开发了自己的编程语言 skill,并为其编写了编译器。由于 skill 语言提供编程接口甚至与 C 语言的接口，所以可以以 Cadence 为平台进行扩展，用户还可以开发自己的基于 Cadence 的工具。实际上，整个 Cadence 软件可以理解为一个搭建在 skill 语言平台上的可执行文件集。所有的 Cadence 工具都是用 Skill 语言编写的，但同时，由于 Cadence 的工具太多，使得 Cadence 显得有点凌乱。这给初学者带来了更多的麻烦。

Cadence 包含的工具较多，几乎包括了 EDA 设计的方方面面。本小册子旨在向初学者介绍 Cadence 的入门知识，所以不可能面面具到，只能根据 ASIC 设计流程，介绍一些 ASIC 设计者常用的工具，例如仿真工具 Verilog-xl,布局布线工具 Preview 和 Silicon Ensemble,电路图设计工具 Composer,电路模拟工具 Analog Artist,版图设计工具 Virtuoso Layout Editor,版图验证工具 Dracula，最后介绍一下 Skill 语言的编程。

1.2 ASIC 设计流程

设计流程是规范设计活动的准则，好的设计流程对于产品的成功至关重要。本节将通过与具体的 EDA 工具（Synopsys 和 Cadence）相结合，概括出

一个实际可行的 ASIC 设计的设计流程。图 1－1 是实际设计过程中较常用的一个流程。



（接下一页）

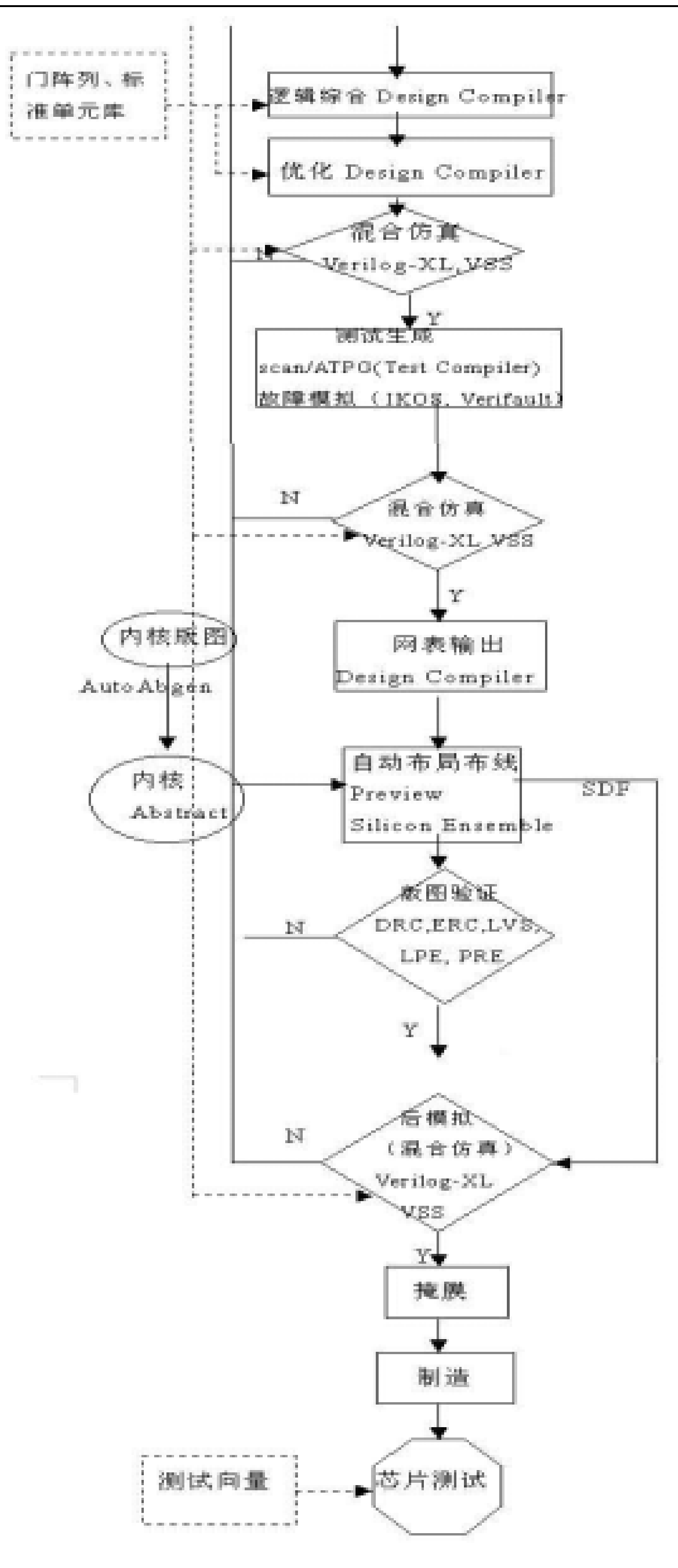


图 1－1 ASIC 设计流程图

这是深亚微米设计中较常用的设计流程。在该设计流程中，高层次综合和底层的布局布线之间没有明显的界线，高层设计时必须考虑底层的物理实现

（高层的划分与布局规划）。同时，由于内核（Core）的行为级模型有其物理实现的精确的延时信息，使得设计者可在设计的早期兼顾芯片的物理实现，从而可以较精确的估计互连的延时，以达到关键路径的延时要求。同时，布局布线后提取的 SDF 文件将被反标到综合后的门级网表中以验证其功能和时序是否正确。

从该流程中可看出，在实际设计中较常用到的 Cadence 的工具有 Verilog HDL 仿真工具 Verilog-XL,电路设计工具 Composer,电路模拟工具 Analog Artist,版图设计工具 Virtuoso Layout Editor,版图验证工具 Dracula 和 Diva 以及自动布局布线工具 Preview 和 Silicon Ensemble。本册子将对这些工具作一个初步介绍。如果读者想进一步了解某个软件的使用，可参考本册子提供的相关在线文档以进一步熟练。

第一章 Cadence 使用基础

2.1 Cadence 软件的环境设置

要使用 Cadence,必须在自己的计算机上作一些相应的设置，这些设置包括很多方面，而且不同的工具可能都需要进行各自的设置。读者如果遇到这方面的问题，可以参考一下 openbook 中的 Configuration Guides 及各工具的 user guide 或者 reference,其访问的方法是 main menu-> System Administration-> Configuration Guides。但作为初学者，只需进行以下几项设置：

1. .cshrc 文件的设置

   首先要在自己的.cshrc 文件中设置 Cadence 软件所在的路径，所使用的 licence 文件等。下面的代码为.cshrc 中设置的一个简单示例，其中 Cadence 所在的目录为/EDA04/cds97a/。

```
##############################
#  #                     Cadence
#
##############################
#
setenv CDS_ROOT /EDA04/cds97a
setenv CDS_INST_DIR /EDA04/cds97a
set path = ($path $CDS_INST_DIR/tools/dfII/bin
        $CDS_INST_DIR/tools/bin)
setenv LM_LICENSE_FILE /EDA04/cds97a/share/license/license.dat
```

   对于某些 Cadence 中的工具也必须在.cshrc 中进行一些设置。

2. .cdsenv 文件设置

   .cdsenv 文件中包含了 Cadence 软件的一些初始设置，该文件是用 Skill 语言写成的。Cadence 可直接执行。

3. .cdsinit 设置

   与.cdsenv 一样，.cdsinit 中也包含了 Cadence 软件的一些初始化设置，该文件是用 Skill 语言写成的。在 Cadence 启动时，会首先自动调用这

两个文件并执行其中的语句。若仅为初学，可以不编写这两个文件，Cadence 会自动调用隐含的设置。若想更改设置，可参考一些模板文件进行编写。在 install_dir/tools/dfII/cdsuser 目录下有一些隐含的模板文件。下面是一个简单的.cdsinit 文件：

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
; Tutorial .cdsinit file
; By:       Cris Reeser/Diane Goldberg
; Created:  October 10, 1995
;
; This initialization file contains the settings necessary to
; successfully run the Cell Design tutorial. Some of these may
; be redundant, if your site uses a site initialization file.
; For further information on initialization files, read the
; comments in the <install_dir>/samples/local/cdsinit file.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
; Human Interface Environment Settings
  hiiSetFont("text" "-adobe-courier-bold-r-*-*-12-*")
  hiSetFormPosition(603:500)
  hinestLimit = 5
  hiSetUndoLimit(10)
  hiExpertMode(nil)
  window(1)->useScrollbars = t
  window(1)->backingStore = t
  envSetVal("layout" "xSnapSpacing" 'float 0.5)
  envSetVal("layout" "ySnapSpacing" 'float 0.5)
  envSetVal("layout" "segSnapMode" 'string "anyAngle")
  envSetVal("layout" "stopLevel" 'int 20)
  envLoadFile("./.cdsenv")
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;
; Bindkey Settings
 load(prependInstallPath("samples/local/schBindKeys.il"))
 load(prependInstallPath("samples/local/leBindKeys.il"))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
; RESIZE CIW
; CIW
; Note, hiFlush() is used as a workaround to display problem
with
; resizing windows in SKILL.
  hiFlush()
  hiResizeWindow(window(1) list(3:3 750:200))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;  Tutorial Customization

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
setSkillPath(". techFiles")
; Welcome the user
 fprintf(poport
"************************************************\n")
 fprintf(poport "Welcome to the SRAM Compiler... %s\n"
getShellEnvVar("USER"))
printf( "    \n" )
printf( "Done with initialization.\n" )
printf("************************************************\n
")
printf( "    \n" )
printf( "    \n" )
```

从中可看出，Skill 语言的语法与 C 语言的较为类似。经过一定的学习后

就很容易掌握。

4. cds.lib 文件的设置

如果用户需要加入自己的库，则可以修改自己的库管理文件 cds.lib。对于初次使用 Cadence 的用户，Cadence 会在用户的当前目录下生成一个 cds.lib 文件，用户通过 CIW 生成一个库时，Cadence 会自动将其加入 cds.lib 文件中。下面是一个简单的 Cadence 库管理文件 cds.lib 的示例：

```
DEFINE ourTechLib
/EDAHOME01/students/dhf/sram/dual/ourTechLib
DEFINE sram /EDAHOME01/students/dhf/sram/dual/sram
DEFINE basic ${CDS_INST_DIR}/tools/dfII/etc/cdslib/basic
DEFINE sample
${CDS_INST_DIR}/tools/dfII/samples/cdslib/sample
DEFINE analogLib
    /EDA04/cds97a/tools/dfII/etc/cdslib/artist/analogLib
DEFINE pCells /EDAHOME01/students/dhf/sram/dual/pCells
DEFINE hhh /EDAHOME01/students/dhf/sram/dual/hhh
```

其中，DEFINE 为库定义的保留字,ourTechLib、sram 等为所定义的库的名字，最后的字符串为保存库的实际的物理目录。

5. 技术库的生成

技术文件库对于 IC 设计而言是非常重要的，其中包含了很多设计中所必需的信息。对于版图设计者而言，技术库就显得更为重要了。要生成技术文件库，必须先编写技术文件。技术文件主要包括层的定义，符号化器件的定义，层、物理以及电学规则和一些针对特定的 Cadence 工具的规则的定义，例如自动布局布线的一些规则，版图转换成 GDSII 时所用到的层号的定义。技术文件的编写可参考 openbook 中有关技术文件的介绍并参考相应的模板来进行。其访问顺序为 Main Menu->IC Tools->Design FramWork II->Technology File Help。附录 1 中有一个简单的技术文件示例。技术文件编好以后，就可以按照以下几步生成技术库：

（1） 点击 CIW 中的 File 菜单选择其中的 New 项中的 Library 项（如图 2－1 所示），弹出图 2－2 所示的表格。

（2）　在 Name 项中输入所需的名字如 myTecLib。保持如图所示的设置，点击 ok。弹出如图 2—3 所示的对话框。

（3）　在对话框中输入编好的技术文件名如 my.tf。这时，技术文件必须在启动 Cadence 的当前目录。点击 ok。

（4）　经过一段时间后，在 CIW 的显示区会出现一个提示

Library myTecLib is created successfully.

对于非工艺库的生成与工艺库大体相同，只是在 2—2 中选择 attach to exited technology file,并在接下来的过程中选择相应的工艺库。

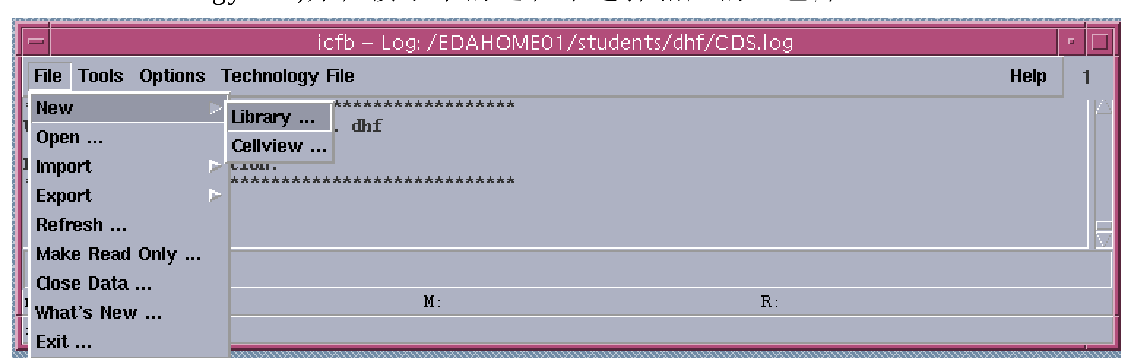


图 2—1

New Library

OK　Cancel　Apply　Help

Library

Name　myTechLib

Directory (non-library directories)

..

/EDAHOME01/students/dhf/tmp

Technology File

If you will be creating mask layout or other physical data in this library, you will need a technology file. If you plan to use only schematic or HDL data, a technology file is not required.

Compile a new techfile

Attach to an existing techfile

Don't need a techfile

Design Manager　No DM

图 2—2

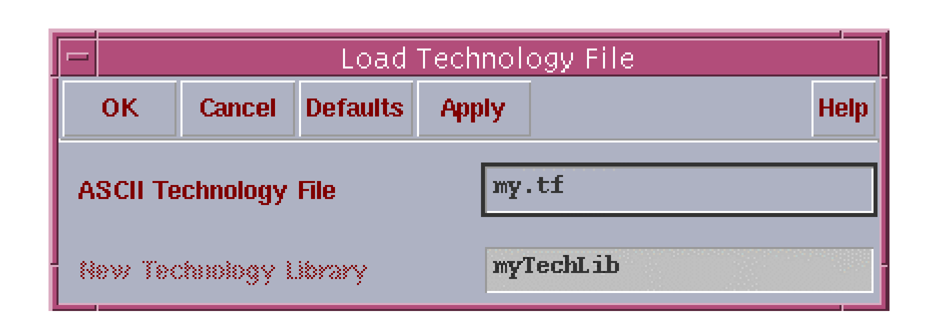


图 2－3

6. 显示文件 display.drf 的设置

   display.drf 文件控制 Cadence 的显示。其基本语法可参考 openbook 中的相应的介绍。附录 1 中包含了一个 display.drf 的示例。

2.2 Cadence 软件的启动方法

完成了一些必要的设置（对初学者只需设置.cshrc 文件即可，其他设置都用隐含设置，等熟练了一些之后，再进一步优化自己的使用环境），就可以启动 Cadence 软件。启动 Cadence 软件的命令有很多，不同的启动命令可以启动不同的工具集，常用的启动命令有 icfb,icca 等，也可以单独启动单个工具，例如启动 Viruoso Layout Editor 可以用 layoutPlus 来启动，Silicon Ensemble 可以用 sedsm 来启动。以 icfb 为例，先在 UNIX 提示符下输入 icfb&，再按回车，经过一段时间，就会出现如图 2－4 所示的 CIW（Command Interpreter Window）窗口。从 CIW 窗口就可以调用许多工具并完成许多任务。

CIW 窗口是使用 Cadence 时遇到的第一个窗口，是 Cadence 主要的用户界面。它主要包括以下几个部分：

1. Title Bar 显示使用的软件名及 log 文件目录。如图 2－4 中的最上一行 icfb-log:/ EDAHOME01/students/dhf/CDS.log。
2. Menu Banner
3. Output Area 输出 Cadence 对用户命令的反应。
4. Input Line 可用来输入 Skill 命令。
5. Mouse Bindings Line 显示捆绑在鼠标左中右三键上的快捷键。
6. Scrolling bar to Scroll Through the Log File

Cadence 将许多常用工具集成在一块以完成一些典型的任务，图 2－5 总结了一些常用的启动命令及其可使用的工具。用户可根据自己的需要选择最少的命令集。

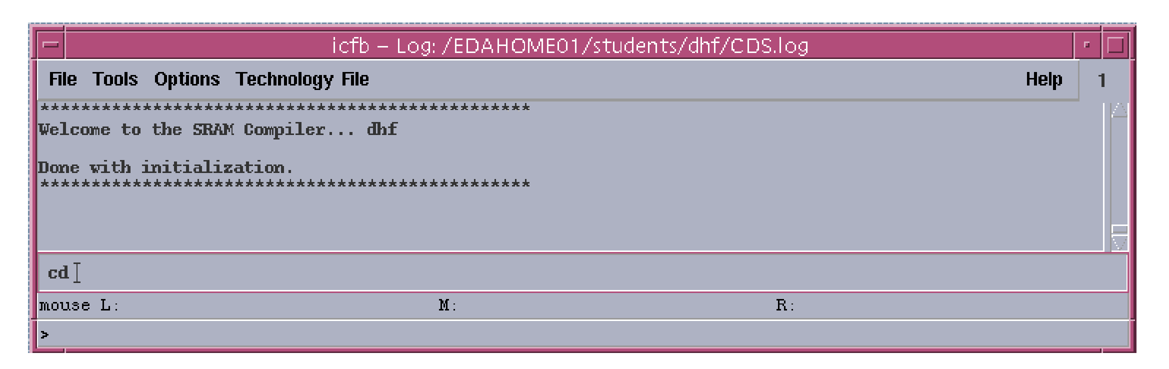


图 2－4 CIW 窗口

	4.4.1 executable	Size	What you might have used before	Type of task you are doing
Front End	*icde*	S	*cds0*	Basic digital and analog design entry
	icds	S	*cds0*	Front end design (*icde* plus digital design environment including PIC, Synergy, TiDS, CSI)
	icms	M	*artist10*	Front end analog, mixed signal, and microwave design (*icds* plus analog, mixed signal and microwave environment and DIVA LVS)
	icca	XL	*cds3*	Front end design with floorplanning (*icds* plus Preview)
Layout	*layout*	S	*cds1*	Basic layout design with interactive DRC
	layoutPlus	M	*cds2*	Basic layout design with automated design tools and interactive verification (*layout* plus LAS, Compactor, DIVA, InQuery, DLE)
Place and Route	*icca*	L	*cds3*	Cell based chip assembly (Preview, Cell Ensemble, Block Ensemble)
Systems	*swb*	S	*swb*	Printed circuit board design (includes interface to Allegro)
	msfb	L	*artist12*	Mixed-Signal IC Design. Excludes Place and Route software
	icfb	XL	*cds3*, *artist13*	Front to back design (includes most Cadence tools; no Dracula or Vampire)

图 2－5 Cadence 启动命令

2.3 库文件的管理

启动了 Cadence 后，就可以利用 File 菜单建立自己的工作库，点击 CIW 窗口上的 File 菜单，选定其中的 New lib 项，弹出如图 2－2 所示的对话框，输入库名并选择相应的工艺库，然后选择 ok,这时在 CIW 的显示区会出现如下提示：

The lib is created successfully!

新建的库是一个空的库，里面什么也没有，用户可在库中生成自己所需的单元。例如可以生成一个反相器单元，并为其生成一个电路及一个版图视图，其流程如下：

1. 选择 File 菜单中的 New 项，并选择 Cellview 项，则弹出如图 2-6 所示的对话框，选择所需的库并输入单元名 inv，并选择视图类型 Schematic，再点击 ok 按钮。则弹出如图 2－7 所示的窗口。
2. 用 Add 菜单中的 Component 命令调用 analogLib 中的单元，输入 PMOS 和 NMOS 管以及电源和地。如图 2－8 所示。
3. 点击 Check and save 命令保存。

用同样的流程可生成 inv 的版图视图。利用 Tools 中的 library manager 可以对库进行管理。

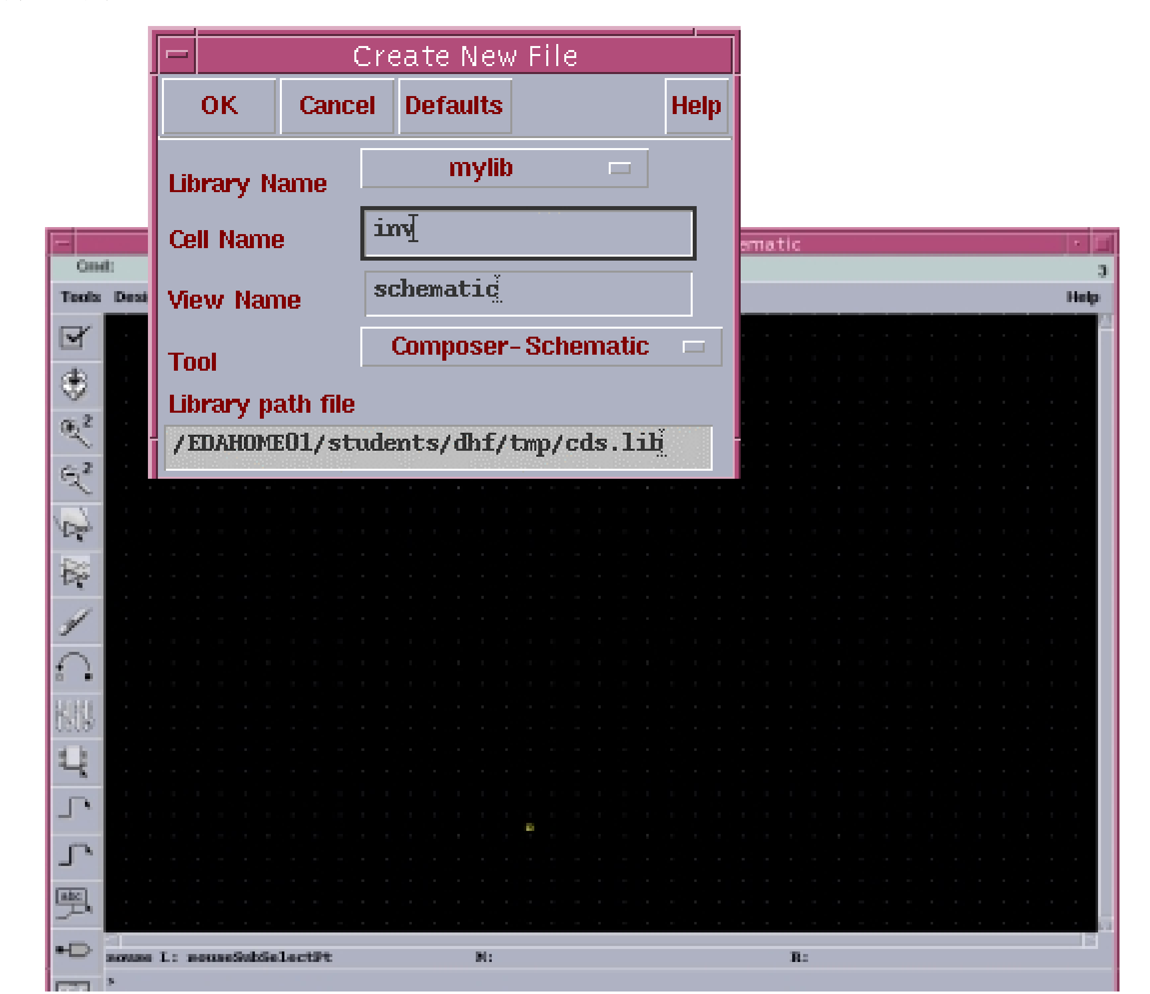


图 2－7

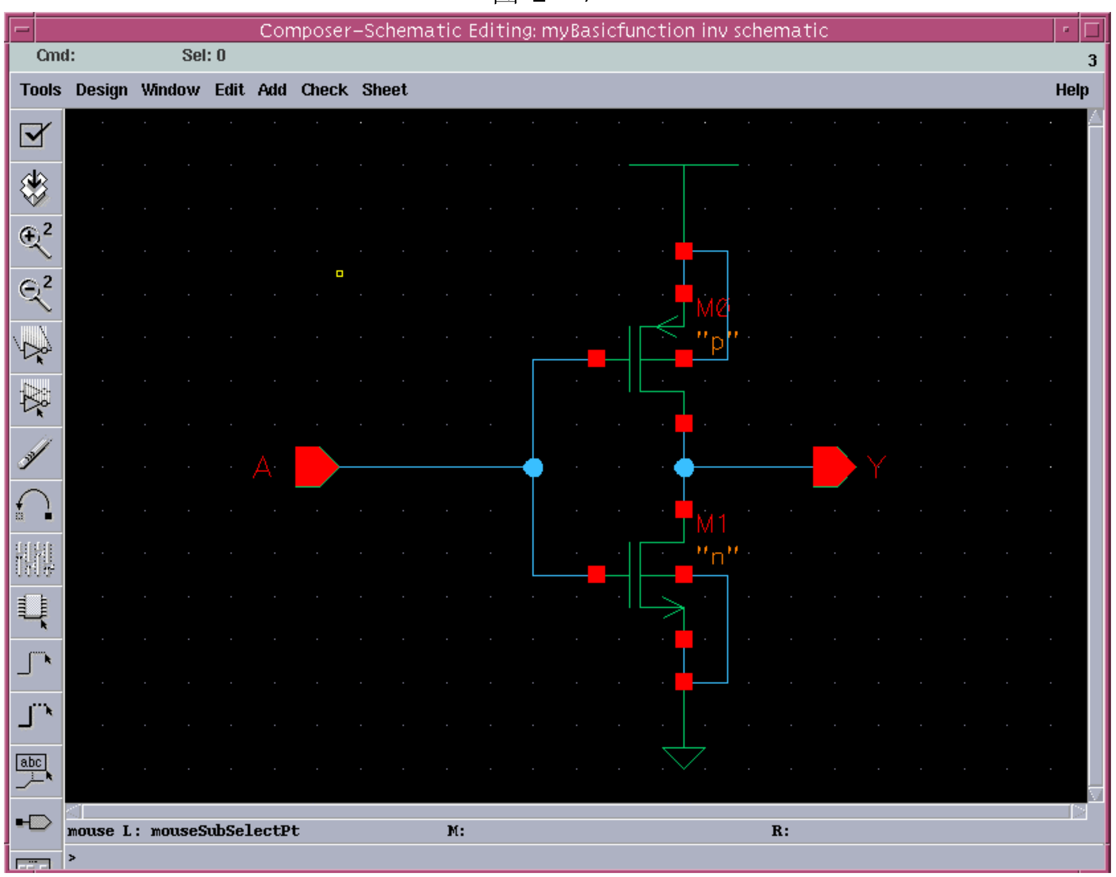


图 2－8

2.4 文件格式的转化

Cadence 有自己的内部数据格式，为了与其他 EDA 软件之间进行数据交换，Cadence 提供内部数据与标准数据格式之间的转换。点击 CIW 的 File 菜单中的 Import 可将各种外部数据格式转换成 Cadence 内部数据格式，利用 CIW 的 File 菜单中的 Export 可将各种 Cadence 内部数据格式转换成外部标准数据格式。

2.5 怎样使用在线帮助

学习 Cadence 的最好教材是使用在线帮助，Cadence 的在线帮助是用 openbook 命令来启动的。在 UNIX 提示符下输入 openbook&并回车就可以启动在线帮助。要拷贝在线帮助中的文件可以先按下 control 键，并用左键进行选择，然后用 copy 进行拷贝。如果想要知道一些关于如何使用 openbook 的技巧，可在系统提示符下输入

openbook help &

即可。

2.6 本手册的组成

在本手册中将按照 ASIC 设计流程分别在第三章介绍高层的 HDL 工具，例如 Verilog 仿真工具 Verilog-xl。第四章介绍电路图设计工具 Composer 及电路模拟工具 Analog Artist。第五章介绍自动布局布线 Preview 和 Silicon Ensemble。第六章介绍版图设计工具 Virtuoso Layout Editor 和验证工具 Dracula 和 Diva。第七章将介绍 Skill 语言的编程。

第二章 Verilog-XL 的介绍

人们在进行电子设计时较常用的输入方法有两种，一种为硬件描述语言，一种为电路图输入。随着 ASIC 设计技术的发展，以 HDL 作为输入的设计方法已成为 ASIC 设计的主流。目前较常用的硬件描述语言有 VHDL 和 Verilog 两种。相对而言，Verilog 在工业上用的较为平常。故本小册子的讨论集中在 Verilog 上。作为 EDA 设计的主流软件之一，Cadence 提供了对 Verilog 及 VHDL 的强大支持。尤其是 Verilog,Cadence 很早就引入了 Verilog,并为其开发了一整套工具。而其中最出色的当数 Verilog 的仿真工具 Verilog-XL。Verilog-XL 一直以其友好的用户界面及强大的功能而受到广大 Verilog 用户的青睐。本章将分五个方面一一对对其进行一个较为详尽的介绍。

3. 1 环境设置

对于一般的 Cadence 的用户而言，可能不需要进行任何设置就可启动 Verilog-XL。用户可输入下列命令看自己是否可访问 Verilog-XL：

which verilog

如果可以访问 Verilog-XL,会有类似如下的反应：

/EDA04/cds97a/ tools/bin/verilog

否则，必须在.cshrc 中用 set path 命令加入以上路径。

3.2 Verilog-XL 的启动

Verilog-XL 的启动命令为 verilog,它可以附带很多可选项，下面是其各选项及其意义：

```
Valid host command options for verilog:
  -f <filename>  read host command arguments from file
  -v <filename>  specify library file
  -y <directory> specify library directory
  -c             compile only
  -s             enter interactive mode immediately
  -i <filename>  input from command file
```

```
  -r <filename>  restart from a saved data structure
  -l <filename>  set log file name
  -k <filename>  set key file name
  -u             convert identifiers to upper case
  -t             set full trace
  -q             quiet
  -d             decompile data structure

Special behavioral performance options (if licensed):
  +turbo         speed up behavioral simulation.
  +turbo+2       +turbo with second level optimizations.
  +turbo+3       +turbo+2 with third level optimizations.
  +listcounts    generate code for maintaining information for
$listcounts
  +no_turbo      don't use a VXL-TURBO license.
  +noxl          disable XL acceleration of gates in all modules

Special environment invocation options (if licensed):
  +gui           invoke the verilog graphical environment
```

下面是几个简单的使用示例，在 UNIX 提示符下输入这些命令即可启动 Verilog-XL：

```
Example host commands to run VERILOG:
  verilog sio85.v
  verilog f1 f2 f3
  verilog -s sio85.v
  verilog -r save.dat -l run2.log -k run2.key
  verilog -r save.dat -si commands.vic
  verilog -dqcr save.dat
```

一般较常用的启动方法是：

```
verilog -s +gui -v libname -f scriptFile sourcefilename &
```

其中，libname 为所使用的库的名字，scriptFile 为用可选项编

写的命令文件。

3.3 Verilog－XL 的界面

运行以上的启动命令后，如果未发生什么错误，就会弹出下图所示的用户界面，这就是 Verilog-XL 的 SimControl 窗口,从该图形界面中，可控制仿真的执行。

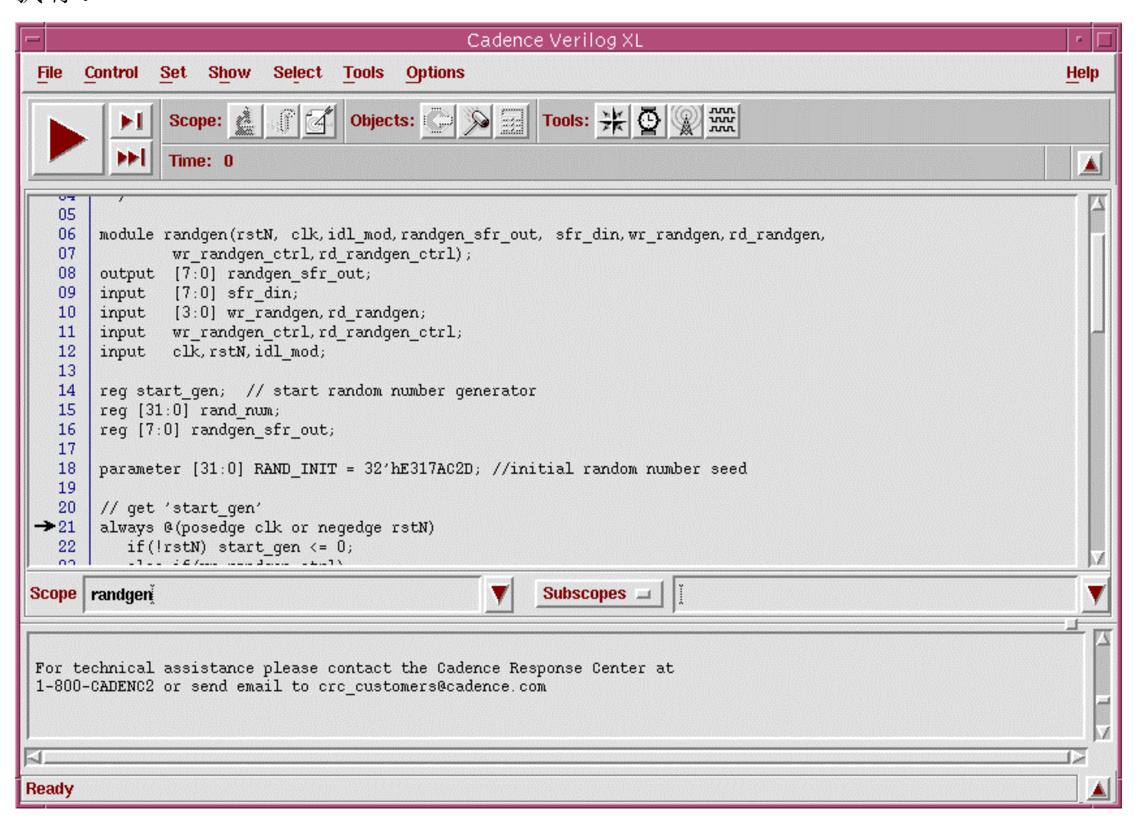


图 3－1 Verilog-XL 的图形界面

Verilog-XL 的图形界面主要有以下几个窗口：

1． SimControl

SimControl 窗口是主要的仿真控制窗口，当用带有＋gui 选项的 verilog 命令启动 Verilog-XL 时，就会弹出这个窗口。通过这个窗口，用户可以显示设计的模块结构，运行 Verilog-XL 命令，设置及显示断点，强行给变量赋值等等。通过这个窗口可以实现用户与仿真的交互，从而达到对仿真的控制。

2． Navigator

通过点击 SimControl 窗口右上角的星形图标即可激活 Navigator 窗口。该窗口可用来图形化显示设计的层次，设计中的实体及其变量。

3． Signal Flow Browser

4． Watch Objects Window

5． SimWave

SimWave 窗口可以用来显示已经选择并跟踪了的信号的波形。

3.4 Verilog-XL 的使用示例

介绍了 Verilog-XL 的启动和用户界面后，下面我们将通过一个具体的实例来演示 Verilog-XL 的使用。在附录 2 中有本示例所需的文件，在本示例中，将对一个 SRAM 模块 SRAM256X8.v 进行仿真。在这个 SRAM 模块中又包含了一个子模块 ram_sy1s_8052.v。所调用的为 TSMC 的 0.35um 的库。test_bench 为 test_memory.v，在该 test_bench 中首先对 SRAM 进行写，然后进行读。下面按照一个简单的流程来对这个 SRAM 进行模拟：

1． 在 UNIX 提示符下输入：

```
verilog -c -v tcb773s.v    test_memory.v &
```

来对源文件进行调试，如果没有错误，会显示 0 Simulation events。

2． 没有错误之后，就可以启动 Verilog-XL 的图形界面：

```
verilog –s +gui –v tcb773s.v   test_memory.v &
```

则会弹出如图 3－2 所示的窗口。

3． 跟踪自己所需要的波形信号。

4． 按运行按钮或在命令行输入原点并回车，即可运行，按停止按钮即可停止。停止后波形会自动更新。

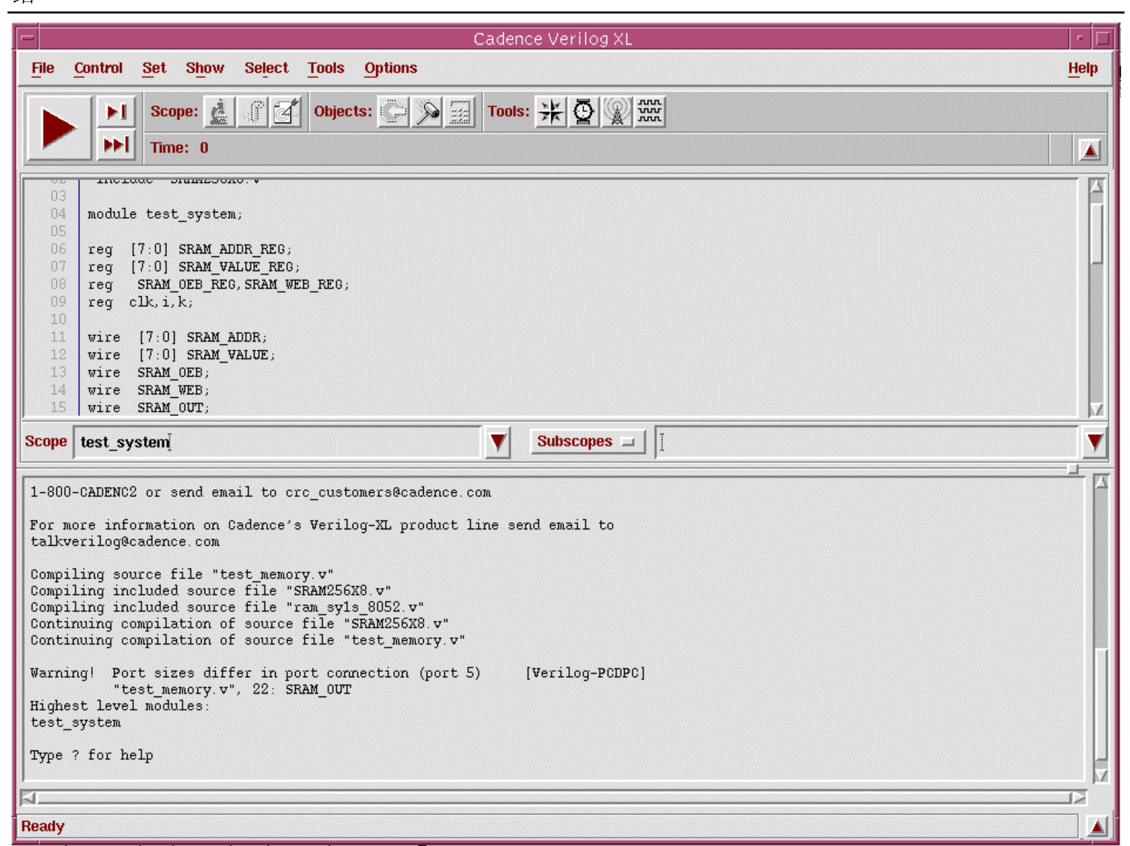


图 3－2

3.5 Verilog-XL 的有关帮助文件

与 Verilog-XL 有关的帮助文件主要有以下一些：

Verilog-XL Reference

Verilog-XL User Guide

Verilog-XL Tutorial

SimCompare User Guide

SimWave User Guide

VPI User Guide and Reference (formerly PLI 2.0)

PLI 1.0 User Guide and Reference

PLI Application Note: Back Annotation and Delay Calculation

PLI Application Note: Using the Value Change Link

LMC Hardware Modeling Interface Reference and User Guide

Graphical Output for the Verilog Product Family Reference

SDF Annotator User Guide

Central Delay Calculator Algorithm Guide

Timing Library Format Reference

Verilog Language Sensitive Editor User Guide

可通过如下顺序对这些文档进行访问：Main menu->HDL Tools－>Verilog-XL。

第四章 电路图设计及电路模拟

设计的输入除了可以用硬件描述语言（如 VHDL 及 Verilog）外，还可以用电路图输入。在早期的 ASIC 设计中，电路图起着更为重要的作用，作为流行的 EDA 软件,Cadence 提供了一个优秀的电路图编辑工具 Composer。Composer 不但界面友好，操作方便，而且功能非常强大。

电路图设计好后，其功能是否正确，性能是否优越，必须通过电路模拟才能进行验证。Cadence 同样提供了一个优秀的电路模拟软件，Analog Artist。由于 Analog Artist 通过 Cadence 与 Hspice 的接口调用 Hspice 对电路进行模拟。本章将介绍电路图设计工具 Composer 和电路模拟软件 Analog Artist 的设置、启动、界面及使用方法、简单的示例以及相关的辅助文件。以便读者能对这两种工具有一个初步的理解。

4.1 电路图设计工具 Composer

Composer 是一种设计输入的工具，逻辑或者电路设计工程师，物理设计工程师甚至 PCB 板设计工程师可以用它来支持自己的工作。

4.1.1 设置

对于一般的 Cadence 的用户而言，可能不需要进行任何设置就可启动 Composer。但有时必须设置快捷键，否则所有的快捷键就会失灵，给使用带来一些不便。在设计时，快捷键往往会有很大的作用。

此外，在电路设计中可能需要用到一些符号库，例如 sample 库，basic 库，analogLib 库，只需在 cds.lib 文件中加入以下一段代码：

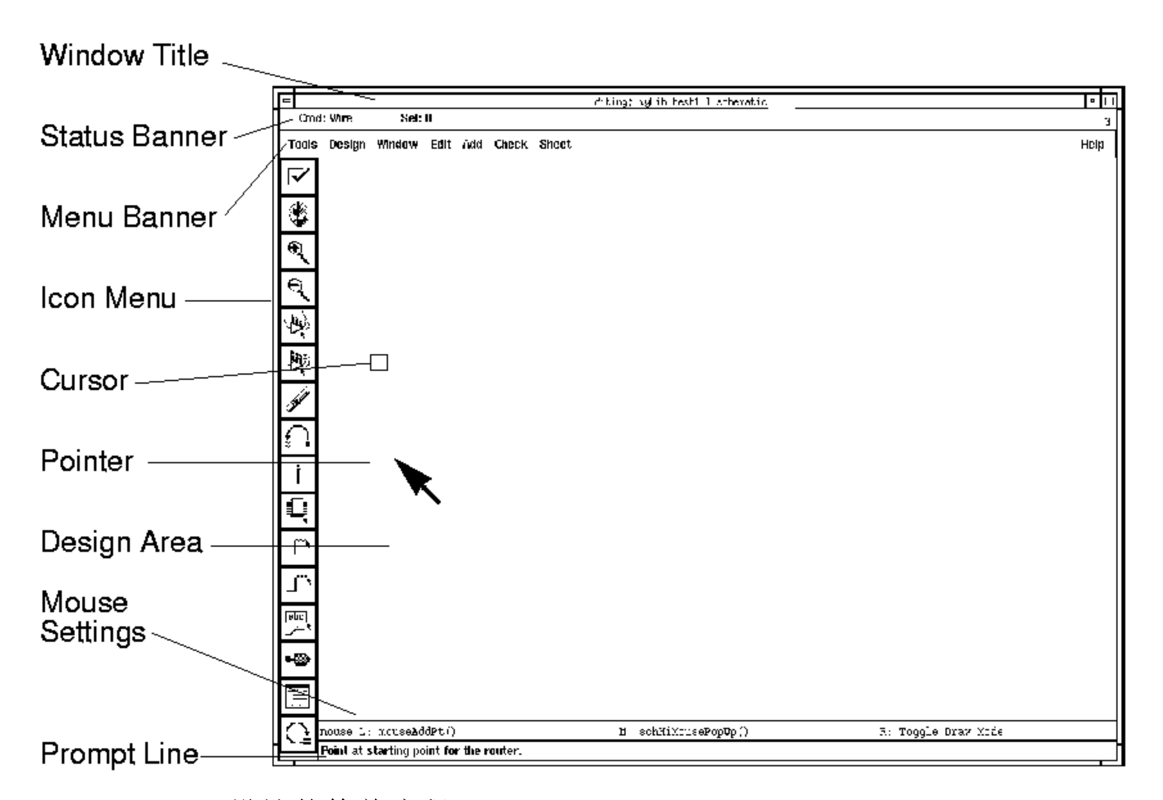
```
DEFINE basic ${CDS_INST_DIR}/tools/dfII/etc/cdslib/basic
DEFINE sample ${CDS_INST_DIR}/tools/dfII/samples/cdslib/sample
DEFINE analogLib
    /EDA04/cds97a/tools/dfII/etc/cdslib/artist/analogLib
```

4.1.2 启动

Composer 的启动很简单，在启动 Cadence 后，从 CIW 窗口中打开或新建一个单元的 Schematic 视图，就会自动启动 Composer 的用户界面。用户即可在其中放入单元及连线，以构成电路图。

4.1.3 用户界面及使用方法

图 4－1 是 Composer 的图形界面，在该用户界面中，大部分面积是右下角的显示区。左边的图标是一些常用的工具，读者可以自己启动 Composer，然后熟悉一下 Composer 的用户界面。下面将简单介绍一下电路图设计及符号



（Symbol）设计的简单流程。

图 4－1 Composer 的用户界面

图 4－2 是编辑电路图的一般流程为：

1. 首先用 Component 命令调用符号库中的元件来添加元件，如图的 nand3。
2. 添加完所有的元件后就可以加入 pin,可通过 add 菜单中的 pin 项来进行添加。
3. 布线及标线名，可通过 wire 命令布线，通过更改其属性标上线名。

4. 添加节点

5. 加注释

6. 加整体属性，如一些自动布局布线属性。

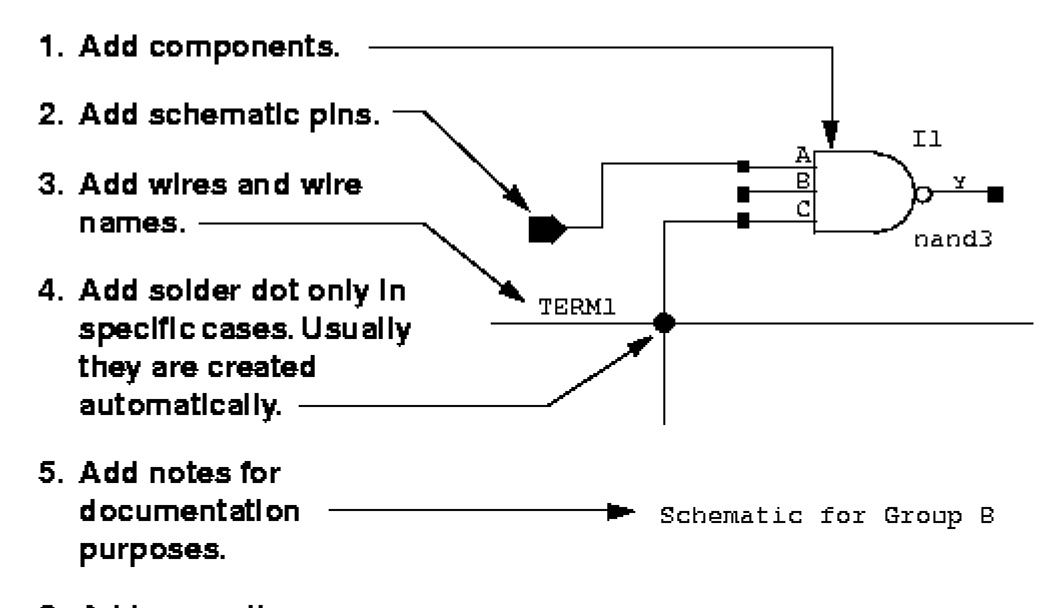


图 4－2 电路图设计的简单流程

符号是用来代表元件的简单符号，如反相器用一个三角形代替。在 Cadence 中，当上层调用下层单元和进行上下级映射时通常调用其符号。所以，符号在电路设计中起着很重要的作用。与启动 Schematic Editor 类似，通过在 CIW 窗口中新建或打开一个单元的 symbol 视图，就可启动 Symbol Editor。图 4－3 是编辑符号的一般流程，主要包括以下几步：

1. 在编辑区加入一些基本的图形。

2. 加入符号的 pin。

3. 加入连接基本图形与 pin 的线。

4. 加入符号的标记，如 inv。

5. 加入选择外框。

6. 加入文本注释

7. 更改整体属性

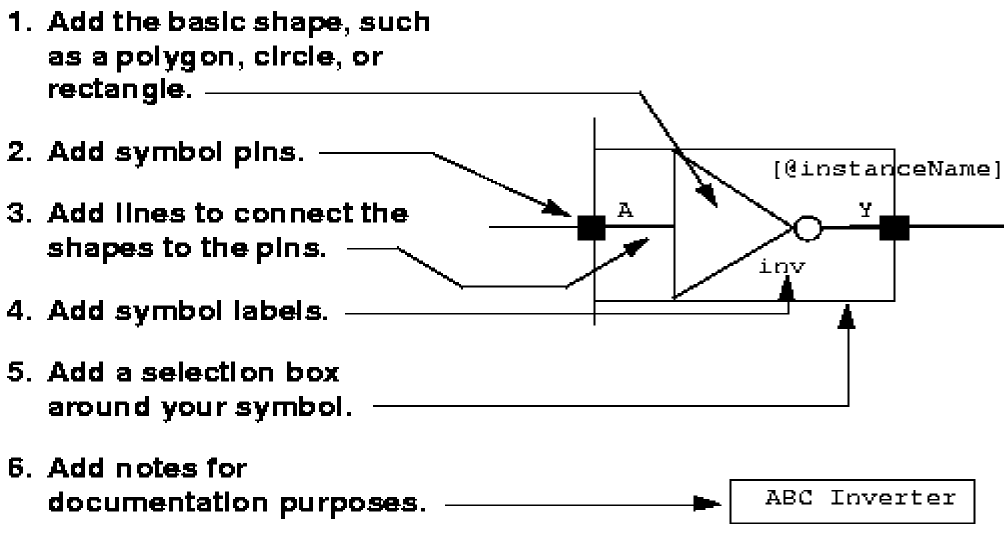


图 4－3 符号设计的简单流程

4.1.4 使用示例

在 openbook 中有一个关于 Composer 的教程，如果读者需要经常用到电路图，本人建议你不妨去走一遍那个教程，对你一定会有帮助的。该教程可安如下顺序进行访问。Main Menu-> IC Tools->Tutorials-> Composer。

4.1.5 相关在线帮助文档

Composer: Design Entry help

4.2 电路模拟工具 Analog Artist

Cadence 提供进行电路模拟的工具 Analog Artist。Anglog Artist 通过调用 Hspice 进行电路模拟，然后进行各种后续处理并显示结果。

4.2.1 设置

在运行 Analog Artist 之前，必须在.cshrc 中设置以下语句：

setenv CDS_Netlisting_Mode Analog

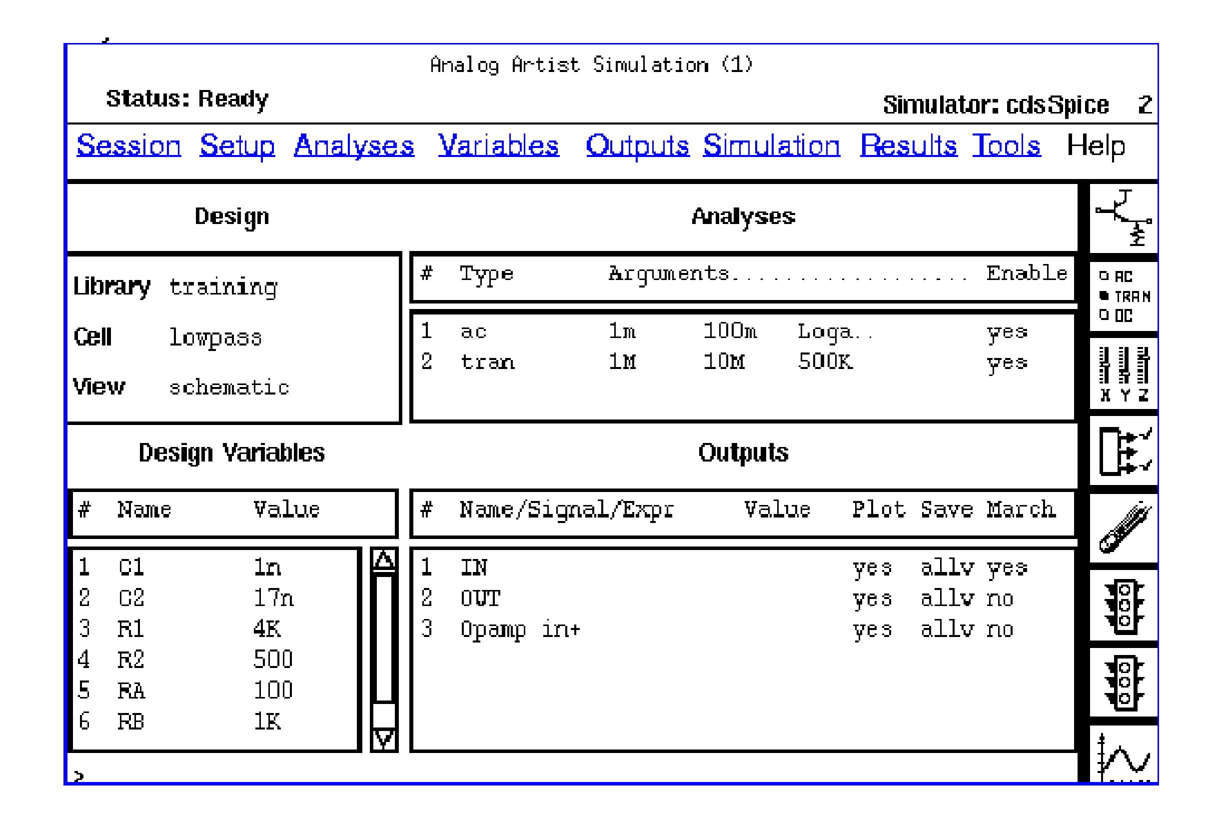
此外，最好能从 Cadence 的安装目录的 Analog Artist 中拷贝与模拟器相应的初始化文件。

4.2.2 启动

Analog Artist 的启动方法有很多种，可以从 Composer 的 Tools 菜单中执行，也可以从 CIW 的 Tools 菜单中执行。

4.2.3 用户界面及使用方法

图 4－4 是 Analog Artist 的用户界面，关于具体的使用方法请参考 openbook 中的相应手册。但有一点想提醒大家，大家使用的 licence 可能不允许使用 Analog Artist。如果在微所使用 Analog Artist 且用 Hspice 为模拟器，似乎激励文件用 cdsspice 格式才可调通，有兴趣的读者可以一试。



4.2.5 相关在线帮助文档

与 Analog artist 相关的在线文档有：

Analog Artist Simulation Help

Analog Artist Microwave Design Help

Analog Artist Mixed-Signal Simulation Help

Analog Artist Parametric Analysis Help

Analog Artist Substrate Coupling Analysis (SCA) Help

Analog Artist SKILL Functions Reference

Analog Expression Language Reference

Cadence SPICE Reference

Component Description Format User Guide

Functional Block Library Reference

HSPICE/SPICE Interface and SPICE 2G.6 Reference

Spectre Reference

Spectre User Guide

SpectreHDL Reference

SpectreRF Help

Switched Capacitor Design System Help

Analog Artist Tutorial: Switched Capacitor Design

Verilog-A Reference

通过顺序 Main Menu-> IC Tools->Analog and Mixed Signal Simulation 可以访问

第五章 自动布局布线

5.1 Cadence 中的自动布局布线流程

从第一章的 ASIC 设计流程中可看到，设计输入经过综合和优化后，就该对所生成的门级网表进行自动布局布线。自动布局布线是连接逻辑设计和物理设计之间的纽带。

在自动布局布线前必须进行布局规划（floorplan）,在 Cadence 中进行布局规划的工具为 Preview，进行自动布局布线的引擎有四种：Block Ensemble、Cell Ensemble、Gate Ensemble 和 Silicon Ensemble。其中，Block Ensemble 适用于宏单元的自动布局布线，Cell Ensemble 适用于标准单元或标准单元与宏单元相混合的布局布线，Gate Ensemble 适合于门阵列的布局布线，Silicon Ensemble 主要用在标准单元的布局布线中。将 Preview 与四种引擎相结合可产生四种不同的自动布局布线环境和流程。由于 Silicon Ensemble(DSM)的功能很完全，几乎可以完成所有复杂的自动布局布线的任务，在考虑自动布局布线引擎时，我们采用了 Silicon Ensemble。SRAM 编译器所生成的用于自动布局布线的端口模型为 Silicon Ensemble 所要求的格式。

图 5－1 为采用 Preview 和 Silicon Ensemble 进行自动布局布线的流程图。该流程主要由以下几个主要步骤组成：

（1） 准备自动布局布线库

在进行自动布局布线之前，必须准备好相应的库。该库中含有工艺数据、自动布局布线用的库单元及显示信息。库的格式必须为 Design Framework II 的数据库格式，可以由用户利用版图生成工具 Virtuoso Layout Editor 设计产生，也可以来自一个由芯片制造厂家和 EDA 公司提供的 LEF(Library Exchange Format)文件，或者从 GDSII 生成。

（2） 准备用来进行自动布局布线的网表

用来进行布局布线的网表可以由硬件描述语言经过综合优化或由电路提取而来。所有网表在进行自动布局布线前都必须首先生成对应的 autoLayout 视图（view)。

（3） 用 Preview 进行布局规划

Preview 是 Cadence 的布局规划器。它可以用来规划物理设计，从而在自动布局布线前预估物理实现的影响。在 Cadence 中，使用 Preview 与自动布局布线引擎相结合来进行自动布局布线。

（4） 用 Silicon Ensemble 进行自动布局布线

（5） 对完成布局布线的版图进行验证

生成的版图其连接性是否正确，是否符合设计规则，是否符合时序要求等等，必须通过验证才能确定。通过点击 Verify&Report 菜单中的相应项，可对版图进行连接性、设计规则验证，并可生成 SDF（Standard Delay Format）文件。通过反标 SDF 文件可对原来的门级网表进行仿真，从而确定其功能和时序是否正确。

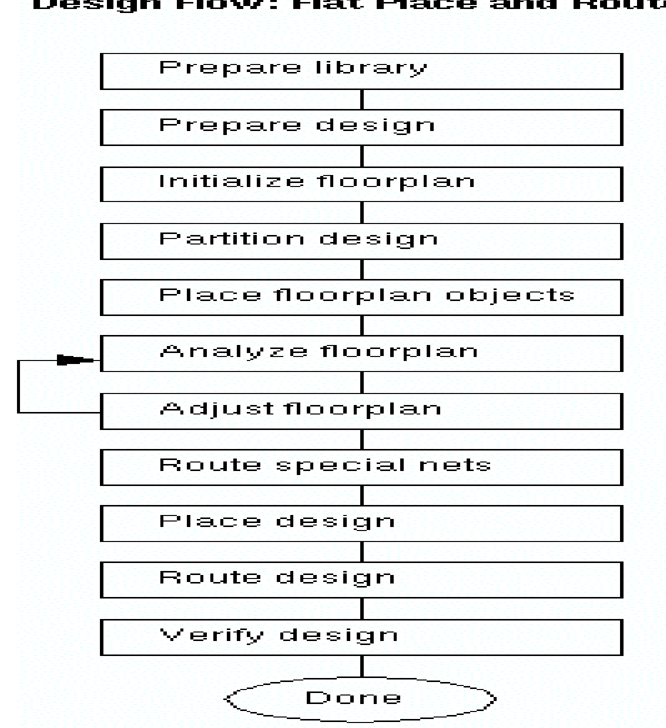


图 5－1 用 Preview 和 Silicon Ensemble 进行自动布局布线的流程

5.2 用 AutoAbgen 进行自动布局布线库设计

对于不同的自动布局布线引擎,对应的库的数据格式有所不同,用来生成库的工具也不同。本 SRAM 编译器选择 Silicon Ensemble 作为布局布线引擎，其对应的库生成工具为 AutoAbgen。AutoAbgen 可以用来生成与用户设计的版图或版图库所对应的 Abstract(即用于自动布局布线的端口模型)。

可以用 AutoAbgen 的 AutoAbgen Flow Sequencer form 来生成 Abstract（对于单个版图）和 LEF 文件（对于整个物理库），其基本流程如下：

(1) 首先在局部.cdsinit 中设置好 AutoAbgen 运行的环境，即在.cdsinit 中加入以下语句：

aabsInstallPath=“<install_dir>/tools/autoAbgen/etc/autoAbgen”

load(buildstring(list(aabsInstallPath “aaicca.ile”) “/”))。

(2) 将 AutoAbgen 的初始化文件.autoAbgen 拷入运行目录。并用 icfb&启动 Cadence。

(3) 点击 CIW 窗口中的 AutoAbgen 菜单下的 AutoAbgen Flow Sequencer 项，打开 Flow Sequencer Form。

(4) 选择合适的流程。

(5) 建立布局布线所需的工艺信息。如果在工艺文件中已经包含布局布线的工艺信息，可以忽略这一步。

(6) 建立用来生成 Abstract 的版图数据。如果所用的版图数据已经是 DFII 的版图格式，可以忽略这一步。

(7) 更新单元的属性及其管脚属性。由于 AutoAbgen 对所操作的版图有些特殊要求，所以在生成 Abstract 前必须对其属性进行更新，以符合 AutoAbgen 的要求。

(8) 建立一个库单元，将所需建立 Abstract 的所有单元包括到里面。

(9) 填写环境设置表格和运行选项表格。输入输出 LEF 的文件名（如果是对库进行操作）。

(10) 选择 Apply 运行 AutoAbgen，生成所需的 Abstract。

第六章版图设计及其验证

如果有人问，“Cadence 最突出的优点在那里？”我想问题的答案应当就在本章。可以说,Cadence 的版图设计及验证工具是任何其他 EDA 软件所无法比拟的。Cadence 的版图设计工具是 Vituoso Layout Editor,即为版图编辑大师，以下简称版图大师。版图大师不但界面很漂亮，而且操作方便，功能强大。可以完成版图编辑的所有任务。

版图设计得好坏，其功能是否正确，必须通过验证才能确定。Cadence 中进行版图验证的工具主要有 Dracula 和 Diva。两者的主要区别是，Diva 是在线的验证工具，被集成在 Design Frame Work II 中，可直接点击版图大师上的菜单来启动。而 Dracula 是一个单独的验证工具，可以独立运行。相比之下，Dracula 的功能比较强大。

6.1 版图设计大师 Virtuoso Layout Editor

版图设计大师是 Cadence 提供给用户进行版图设计的工具。其使用起来十分方便，下面进行一个简单介绍：

6.1.1 设置

版图大师的设置很简单，对于一般的 Cadence 的用户而言，可能不需要进行任何设置就可启动版图大师。但有时必须设置快捷键，否则所有的快捷键就会失灵，给使用带来一些不便。在设计时，快捷键往往会有很大的作用。

与电路设计不同的是，版图设计必须考虑具体的工艺实现，因此，存放版图的库必须是工艺库或附在别的工艺库上的库。否则，用隐含的库将没有版层，即 LSW 窗口只有一个黑框，更无从画图了。因此，在设计版图前必须先建立自己的工艺库。

此外，显示对于版图设计也很重要，因此最后有自己的显示文件 display.drf。

6.1.2 启动

有很多种方法自动版图大师，最简单的办法是通过 CIW 打开或者新建一

个单元的版图视图，这样就会自动启动版图大师。此外，也可以用 layoutPlus 或 layout 命令启动。

6.1.3 用户界面及使用方法

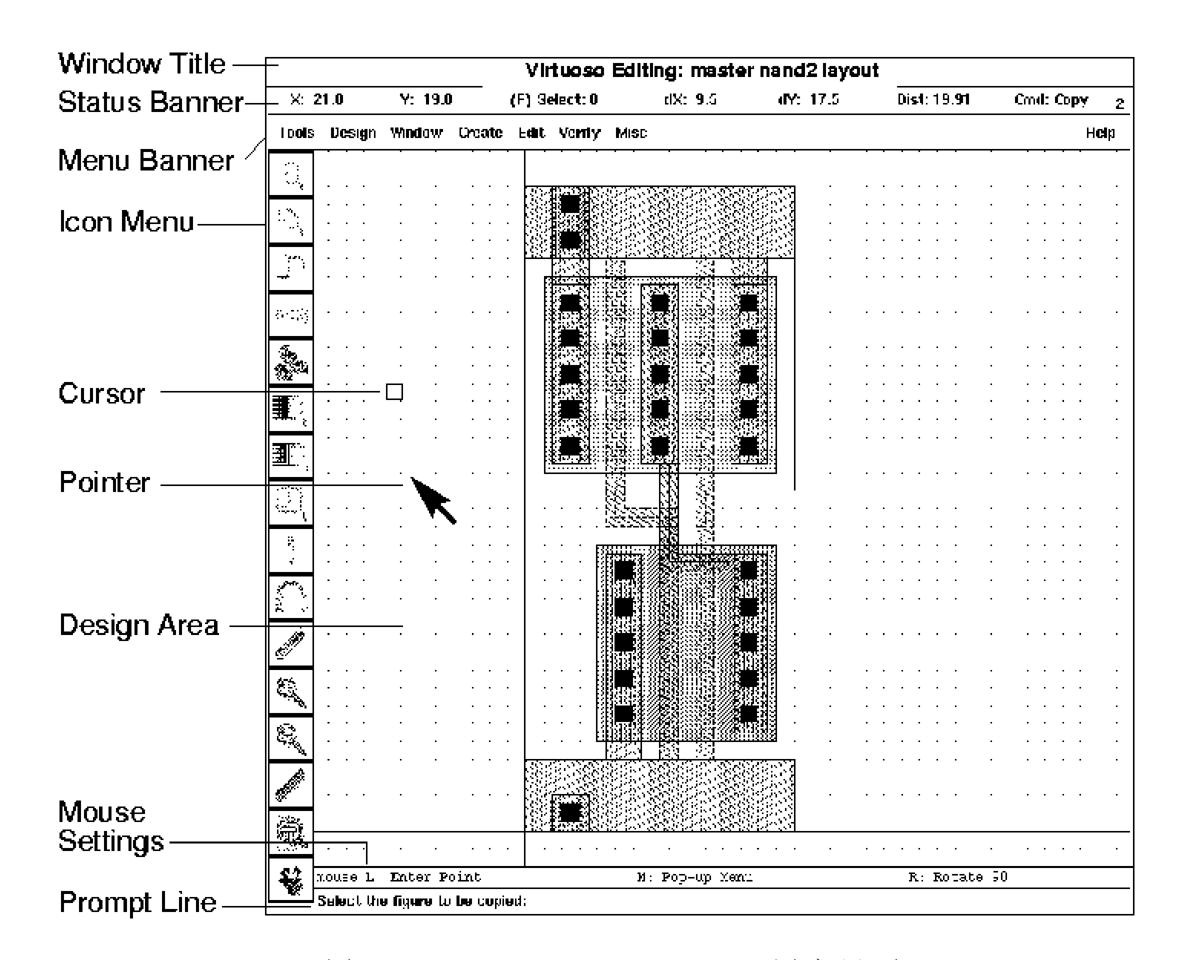


图 6－1 Virtuoso Layout Editor 用户界面

通过上述方法启动版图大师后，就会出现如图 6－1 所示的用户界面及一个 LSW 窗口。从 LSW 窗口中选择所需的层，然后在显示区画图。具体的操作请读者参考 openbook 中的部分。

6.1.4 使用示例

关于 Virtuoso Layout Editor 的具体使用，在此就不再赘述，在 openbook 中有一个很好的例子 cell_design。建议所有学习 Cadence 的人都该走一遍该教程。该教程的访问顺序为： Main Menu->IC Tools->Tutorials and Flow Guides-> Cell Design Tutorial。

6.1.5 相关在线帮助文档

与版图大师有关的在线文档有：

Virtuoso Layout Editor help

Cell Design Tutorial

6.2 版图验证工具 Dracula

6.2.1 Dracula 使用介绍

用 Virtuoso Layout Editor 编辑生成的版图是否符合设计规则、电学规则，其功能是否正确必须通过版图验证系统来验证。Cadence 提供的版图验证系统有 Dracula 和 Diva。Diva 嵌入在 Cadence 的主体框架之中，使用较方便，但功能较之 Dracula 稍有逊色。Dracula 为独立的版图验证系统，可以进行 DRC（设计规则检查）、ERC（电学规则检查）、LVS（版图和电路比较）、LPE（版图寄生参数提取）、PRE（寄生电阻提取），其运算速度快，功能强大，能验证和提取较大的电路，本手册着重介绍 Dracula 的使用。

使用 Dracula 和 Diva 的第一步是编写与自己的工艺一致的命令文件，包括 DRC、ERC、LVS、LPE 甚至 PRE 文件。关于命令文件的编写，读者可参考 openbook 中的手册，可安以下顺序找到。附录 3 中有一套 0.5um 的命令文件，读者可参考。

假设要验证的版图为 mySRAM 库中的 sram256x8 单元，用来进行验证的当前目录为 myver，运行 Dracula 的命令文件为 mydrc.com。执行 DRC、ERC 和 LPE 的流程如下：

（1） 利用 Virtuoso Layout Editor 生成所需的版图 sram256x8，然后利用 CIW 窗口中的 Export->Stream 菜单，将单元 sram256x8 的版图转变成 GDSII 格式文件 sram256x8.gds，并存到运行目录 myver 下。

（2） 修改运行 Dracula 所需的命令文件 mydrc.com，将其中的 INDISK 文件改为 sram256x8.gds,OUTDISK 改为任何自己喜欢的文件，例如 sram256x8_out.gds，将 WORK-DIR 改为当前的运行目录 myver，将 PRIMARY 改为大写的单元名，即 SRAM256X8。

（3） 在当前目录下运行 PDRACULA，即在 UNIX 操作符下输入 PDRACULA&，然后输入/GET mydrc.com 并回车，接着输入/fi 即可生

成 jxrun.com 及 jxsub.com。

（4） 在当前目录下运行 jxrun.com 或 jxsub.com。

（5） 检查结果文件，DRC 检查为 printfile_name.drc，ERC 为 printfile_name.erc，LVS 为 printfile_name.lvs。其中，printfile_name 为命令文件中 PRINTFILE 所指定的字符串。

（6） 利用 InQuery&命令启动图形界面查找并修改错误。

（7） 重复（1）至（6），直至改完所有的错误。

由于 Dracula 的功能强大，速度较快，可以对整个 SRAM 版图进行验证，所以可以确保生成的 SRAM 版图完全符合设计规则、电学规则。

6.2.2 相关在线帮助文档

与版图验证有关的在线文档主要有以下几个。InQuery 是用来显示验证结果的。

Diva Interactive Verification Reference

Dracula Standalone Verification Reference

Dracula User Guide

InQuery Reference

InQuery Tutorial

第七章 skill 语言程序设计

7.1 skill 语言概述

Skill 语言是 Cadence 公司自己开发的一种类似于 leap 语言的编程语言。Cadence 公司不仅开发了全套的 Skill 语言语法，还开发了一个完整的 Skill 语言的编译器。整个 Cadence 软件都是基于 Skill 语言的，所有的 Cadence 的工具都是用 Skill 语言编写的，甚至各种设置辅助文件都是遵从 Skill 语言的语法。此外，Cadence 公司为其每个产品都提供 Skill 语言访问函数，这使得用户可以通过 Skill 语言访问 Cadence 的产品。由于 Skill 由于提供编程接口及与 C 语言的接口，这使得 Cadence 可以在原有的基础上进行各种扩展，甚至允许用户开发自己的基于 Cadence 平台的工具。因此，学习一些基本的 Skill 语言的知识及基本的编程技巧对于学习 Cadence 是大有益处的。本章将对 Skill 语言作一些简单的介绍。

7.2 skill 语言的基本语法

关于 Skill 语言的基本语法，读者可参考 openbook 中的 Skill Language User Guide 以及 Skill Language Functions Reference 两本手册。可按顺序 Main Menu->Skill Language 找到。

7.3 Skill 语言的编程环境

最简单的 Skill 编程方法是利用 CIW 窗口中的命令解释行，可以将其看为 Skill 语言的行编译器。例如在其中输入

printf(“HI”);

再回车，在 CIW 的显示区就会出现 HI 字符串。除了输入单行的 Skill 命令外，也可先用任一文本编辑器将一系列命令组成一个 skill 文件，例如 myskill.il，然后利用 load 命令将文件导入 Skill 编译器，具体方法为在 CIW 命令解释行中输入 load(“myskill.il”)。如果文件不在当前目录，还需在文件名前加上路径。如果为了保密，可以将 Skill 文件存成 context 的形式。

实际上，Cadence 提供了一套完整的编程工具，在 CIW 的 Tools 菜单下的 Skill

Development 可启动该编程环境。如果读者有兴趣，可自己去摸索，在此就不在赘述了。

7.4 面向工具的 skill 语言编程

Cadence 公司在开发自己的产品时，一般都提供其相应的 Skill 语言函数，产品种的点击菜单等动作都可用执行其相应的 skill 语言函数来代替。这就该编程者提供了宝贵的接口。由于 Skill 由于是 script 语言，因此，熟练的用户可以通过适当的编程实现任务的自动化。下面以面向用户界面的编程为例简单介绍一下 Skill 语言的编程。

下面这段代码是本人编写的一个用户界面，其中最主要用到的是 Cadence 提供的关于用户界面设计的函数，读者可以从中研究一下。图 7－1 至 7－6 是其对应的图形界面。

Skill 程序代码

```
;SRAM Compiler main function
;Define some global variable and the default value
MaxWords = 4096
MaxBits  = 64
MinWords = 192
MinBits  = 1
ModuleName = "SRAM256x8"
SRAMWords = 256
SRAMBits  = 8
SRAMPorts = 1
RowCellNum = 8
ColCellNum = 8
RowAddNum = 4
ColAddNum = 2
ColMul = 4; The multiplier of the bits
PlugNum = 8
LibraryName = "mySRAM"
```

```
TecLibraryName = "ourTechLib"
Workpath = "."


procedure( mainOpen()
           println("I love you!")
    )
procedure( mainLoadState()
           println("I love you!")
        )
procedure( mainSaveState()
          println("I love you!")
        )
procedure( mainReset()
           println("I love you!")
        )
procedure( mainReport()
          view("./sram_compiler/sram.rep")
        )
procedure( mainAbout()
          view("./sram_compiler/sram.ver")
        )
procedure( mainExit()
           hiCloseWindow(w)
       println("Thank you for your use of our SRAM Compiler, Bye:)")
        )
procedure( mainWordBits()
          ;;; creating the words field
    WordsField = hiCreateIntField(
    ?name      'WordsField
    ?prompt    "SRAM Words(64-4096)"
```

```
?value      SRAMWords
?defValue  256
?callback       "SetWordsCB( hiGetCurrentForm() ) "
?range '(64 4096)
)
 BitsField = hiCreateIntField(
     ?name           'BitsField
     ?prompt         "SRAM Bits(1-64)"
     ?value         SRAMBits
     ?defValue      16
     ?callback       "SetBitsCB( hiGetCurrentForm() ) "
     ?range '(1 64)
     )

;;; creating the form
SetWordsForm = hiCreateAppForm(
?name          'SetWordsForm
?formTitle     "Set Words&Bits Form"
?callback      "SetWordBitsFormCB( hiGetCurrentForm() ) "
?fields        list( WordsField BitsField )
?unmapAfterCB                 t
)
   procedure( SetWordsCB( theForm )
        SRAMWords = theForm->WordsField->value
) ; procedure

procedure( SetBitsCB( theForm )
                    SRAMBits = theForm->BitsField->value
    ) ; procedure
```

```
    procedure( SetWordBitsFormCB( theForm )
        if( SetWordsCB( theForm )
            then
                hiSetCallbackStatus( theForm t )
                hiHighlightField( theForm 'WordsField
                'background )
;       view( theForm->WordsField->value )
;           SRAMWords = theForm->WordsField->value
        else
            if( SetBitsCB( theForm )
                                then
                                        hiSetCallbackStatus( theForm t )
                                          hiHighlightField(        theForm
'BitsField 'background )
                        else

            hiSetCallbackStatus( theForm nil )
            hiHighlightField( theForm 'WordsField
                'highlight )
            )
        ) ; if
        ) ; procedure
    ;;; displaying the form
    hiDisplayForm( SetWordsForm )
        )

procedure( mainModuleName()
    ;;; creating the File Name field
    ModuleNameField = hiCreateStringField(
    ?name       'ModuleNameField
    ?prompt     "Module Name"
```

```
    ?value      ModuleName
    ?defValue  "SRAM256x8"
    ?callback  "ModuleFieldCheckCB( hiGetCurrentForm() )"
    ?editable t
    )
        procedure( ModuleFieldCheckCB( theForm )
;   if( isFile( theForm->trFileNameField->value )
;       then
            ModuleName = theForm->ModuleNameField->value
             t
;       else
;           println("File Does Not Exist--Try Again")
;           nil
;       ) ;if
    ) ; procedure

    ;;; creating the form
    ModuleNameForm = hiCreateAppForm(
    ?name          'ModuleNameForm
    ?formTitle             "Module Name Form"
    ?callback              'ModuleNameFormCB
    ?fields        list( ModuleNameField )
    ?unmapAfterCB                t
    )
    procedure( ModuleNameFormCB( theForm )
    if( ModuleFieldCheckCB( theForm )
        then
            hiSetCallbackStatus( theForm t )
            hiHighlightField( theForm 'ModuleNameField
'background )
;           view( theForm->ModuleNameField->value )
```

```
        else
            hiSetCallbackStatus( theForm nil )
            hiHighlightField( theForm 'ModuleNameField
'highlight )
        ) ; if
    ) ; procedure
    ;;; displaying the form
    hiDisplayForm( ModuleNameForm )

        )
procedure( mainLibrary()
        ;;; creating the File Name field
        LibraryNameField = hiCreateStringField(
        ?name           'LibraryNameField
        ?prompt         "Working Library Name"
        ?value          LibraryName
        ?defValue       "mySRAM"
        ?callback       "LibraryFieldCheckCB( hiGetCurrentForm() )"
        ?editable t
        )
        procedure( LibraryFieldCheckCB( theForm )
;       if( isFile( theForm->trFileNameField->value )
;               then
                    LibraryName = theForm->LibraryNameField->value
                        t
;               else
;                       println("File Does Not Exist--Try Again")
;                       nil
;               ) ;if
        ) ; procedure
```

```
        ;;; creating the form
        LibraryNameForm = hiCreateAppForm(
        ?name                   'LibraryNameForm
        ?formTitle                                "Working Library Name
Form"
        ?callback                              'LibraryNameFormCB
        ?fields                 list( LibraryNameField )
        ?unmapAfterCB                                     t
        )
        procedure( LibraryNameFormCB( theForm )
        if( LibraryFieldCheckCB( theForm )
                then
                        hiSetCallbackStatus( theForm t )
                        hiHighlightField( theForm 'LibraryNameField
'background )
;                       view( theForm->ModuleNameField->value )
                else
                        hiSetCallbackStatus( theForm nil )
                        hiHighlightField( theForm 'LibraryNameField
'highlight )
                ) ; if
        ) ; procedure
        ;;; displaying the form
        hiDisplayForm( LibraryNameForm )
    mySRAM = dbCreateLib( LibraryName  Workpath)



        )

procedure( mainTecLibrary()
       ;;; creating the File Name field
```

```
        TecLibraryNameField = hiCreateStringField(
        ?name           'TecLibraryNameField
        ?prompt         "Technology Library Name"
        ?value          TecLibraryName
        ?defValue       "outTechLib"
        ?callback       "TecLibraryFieldCheckCB( hiGetCurrentForm() )"
        ?editable t
        )
        procedure( TecLibraryFieldCheckCB( theForm )
;       if( isFile( theForm->trFileNameField->value )
;               then
                   TecLibraryName = theForm->TecLibraryNameField->value
                       t
;               else
;                       println("File Does Not Exist--Try Again")
;                       nil
;               ) ;if
        ) ; procedure

        ;;; creating the form
        TecLibraryNameForm = hiCreateAppForm(
        ?name                   'TecLibraryNameForm
        ?formTitle                              "Technology Library Name
Form"
        ?callback                               'TecLibraryNameFormCB
        ?fields                 list( TecLibraryNameField )
        ?unmapAfterCB                                   t
        )
        procedure( TecLibraryNameFormCB( theForm )
        if( TecLibraryFieldCheckCB( theForm )
                then
```

```
                        hiSetCallbackStatus( theForm t )
                        hiHighlightField( theForm 'TecLibraryNameField
'background )
;                       view( theForm->ModuleNameField->value )
                else
                        hiSetCallbackStatus( theForm nil )
                        hiHighlightField( theForm 'TecLibraryNameField
'highlight )
                ) ; if
        ) ; procedure
        ;;; displaying the form
        hiDisplayForm( TecLibraryNameForm )



        )

procedure( mainWorkpath()
       ;;; creating the File Name field
        WorkpathField = hiCreateStringField(
        ?name           'WorkpathField
        ?prompt         "Working Path"
        ?value          Workpath
        ?defValue       "."
        ?callback       "WorkpathFieldCheckCB( hiGetCurrentForm() )"
        ?editable t
        )
        procedure( WorkpathFieldCheckCB( theForm )
;       if( isFile( theForm->trFileNameField->value )
;               then
                   Workpath = theForm->WorkpathField->value
                        t
```

```
;               else
;                       println("File Does Not Exist--Try Again")
;                       nil
;               ) ;if
        ) ; procedure

        ;;; creating the form
        WorkpathForm = hiCreateAppForm(
        ?name                   'WorkpathForm
        ?formTitle                              "Working Path Form"
        ?callback                               'WorkpathFormCB
        ?fields                 list( WorkpathField )
        ?unmapAfterCB                                   t
        )
        procedure( WorkpathFormCB( theForm )
        if( WorkpathFieldCheckCB( theForm )
                then
                        hiSetCallbackStatus( theForm t )
                        hiHighlightField( theForm 'WorkpathField
'background )
;                       view( theForm->ModuleNameField->value )
                else
                        hiSetCallbackStatus( theForm nil )
                        hiHighlightField( theForm 'WorkpathField
'highlight )
                ) ; if
        ) ; procedure
        ;;; displaying the form
        hiDisplayForm( WorkpathForm )

mySRAM = dbCreateLib( LibraryName  Workpath)
```

```
        )

procedure( mainPorts()
    PortsField = hiCreateIntField(
    ?name       'PortsField
    ?prompt     "SRAM Ports(1,2,4)"
    ?value      SRAMPorts
    ?defValue   1
    ?callback        "SetPortsCB( hiGetCurrentForm() ) "
    ?range '(1 4)
    )
    ;;; creating the form
    SetPortsForm = hiCreateAppForm(
    ?name           'SetPortsForm
    ?formTitle      "Set Ports Form"
    ?callback       "SetPortsFormCB( hiGetCurrentForm() ) "
    ?fields         list(PortsField)
    ?unmapAfterCB                   t
    )
    procedure( SetPortsCB( theForm )
                    SRAMPorts = theForm->PortsField->value
        ) ; procedure
        procedure( SetPortsFormCB( theForm )
                if( SetPortsCB( theForm )
                        then
                                hiSetCallbackStatus( theForm t )
                                hiHighlightField( theForm 'PortsField
                                'background )
;               view( theForm->WordsField->value )
;                       SRAMWords = theForm->WordsField->value
                ) ; if
```

```
                ) ; procedure
        ;;; displaying the form
        hiDisplayForm( SetPortsForm )
        )



procedure( mainOther()
    ;;; creating the list of items in the cyclic field
printf("It is used to extend the setting")
/*trCyclicList = '(
    "1" "2" "3" "4" "5" "6" "7" "8" "9" "10"
    "11" "12" "13" "14" "15" "16" "17" "18" "19" "20" "21" "22"
    )

;;; creating the cyclic field
trCyclicField = hiCreateCyclicField(
    ?name       'trCyclicField
    ?prompt     "Cycle Through: "
    ?choices    trCyclicList
    ?value      "3"
    ?defValue   "7"
    ?callback   "println"
    )
;;; creating the boolean button field.
;;; The callback for the trBooleanButton field dynamically ;;; retrieves the
new value of the field and embeds it in a ;;; message
trBooleanButton =
    ?name       'trBooleanButton
    ?buttonText    "Boolean"
    ?value      t
    ?defValue   nil
```

```
    ?callback  "println( hiGetCurrentForm()->trBooleanButton-> value )"
    )

;;; creating the button box field
trButtonBoxField =
    ?name       'trButtonBoxField
    ?prompt     "Button Box"
    ?choices    '("Do a" "Do b" "Do c")
    ?callback   '("println( 'a )" "println( 'b )" "println( 'c
)" )
    )

;;; creating the radio field.
;;; The callback for the trConeRadioField field dynamically ;;; retrieves
the new value of the field and imbeds it in a ;;; message
trConeRadioField = hiCreateRadioField(
    ?name       'trConeRadioField
    ?prompt     "Cone Size: "
    ?choices    list( "small" "medium" "large" )
    ?value      "small"
    ?defValue   "large"
    ?callback
        '(  "printf(    \"\n%s  cone  chosen  \"  hiGetCurrentForm()  -
>trConeRadioField->value )" )
    )

;;; creating the scale field
trScaleField =
    ?name       'trScaleField
    ?prompt     "Slide "
    ?value      500
```

```
    ?defValue  250
    ?callback  "println(\"scale changed \")"
    ?range     0:750
    )

;;; creating the label field
trLabelField =
    ?name      'trLabelField
    ?labelText "Label"
    ?justification    CDS_JUSTIFY_RIGHT
    )

;;; creating the form
hiCreateAppForm(
    ?name      'trSampleForm
    ?formTitle "Other Setting Form"
    ?callback  "println( 'FormAction )"
    ?fields
        list(
            trCyclicField
            trBooleanButton
            trButtonBoxField
            trConeRadioField
            trScaleField
            trLabelField
            )
    ?unmapAfterCB  t
    ) ; hiCreateAppForm

;;; displaying the form
hiDisplayForm( trSampleForm )
```

```
*/
    )
procedure( main0Layout()
    case( SRAMPorts
        ( 1
                    load("./sram_compiler/layout.il")
        )
        (2
            load("./sram_compiler/layout_dual.il")
        )
        (4
        println("I love you!")
        )
        ( t
            println("Ports is out of range!")

        )
    )
        )
procedure( main0Abstract()
    load("./sram_compiler/abstract.il")
    )
procedure( main0Symbol()
     case( SRAMPorts
                ( 1
                    load("./sram_compiler/symbol.il")
                )
                (2
                    load("./sram_compiler/symbol_dual.il")
                )
                (4
```

```
                println("I love you!")
                )
                ( t
                        println("Ports is out of range!")

                )
        )

        )
procedure( mainOVerilog()
        load("./sram_compiler/verilog.il")
    )
procedure( mainOTestVec()
        load("./sram_compiler/testVec.il")
    )
procedure( mainOVhdl()
       load("./sram_compiler/vhdl.il")
    )
procedure( mainCloseProc()
           println("Thank you for your use of our SRAM Compiler!!!")
        )
Open_item = hiCreateMenuItem(
    ?name 'Open_item
    ?itemText "Open"
    ?callback "mainOpen()"
    )
Report_item = hiCreateMenuItem(
    ?name 'Report_item
    ?itemText "Result Report"
    ?callback "mainReport()"
    )
```

```
About_item = hiCreateMenuItem(
    ?name 'About_item
    ?itemText "About SRAM Compiler"
    ?callback "mainAbout()"
    )
LoadState_item = hiCreateMenuItem(
        ?name 'LoadState_item
        ?itemText "Load State"
        ?callback "mainLoadState()"
        )
SaveState_item = hiCreateMenuItem(
        ?name 'SaveState_item
        ?itemText "Save State"
        ?callback "mainSaveState()"
        )
Reset_item = hiCreateMenuItem(
        ?name 'Reset_item
        ?itemText "Open"
        ?callback "mainReset()"
        )
Exit_item = hiCreateMenuItem(
        ?name 'Exit_item
        ?itemText "Exit"
        ?callback "mainExit()"
        )
ModuleName_item = hiCreateMenuItem(
        ?name 'ModuleName_item
        ?itemText "Module Name"
        ?callback "mainModuleName()"
        )
Words_item = hiCreateMenuItem(
```

```
        ?name 'Words_item
        ?itemText "Words&Bits"
        ?callback "mainWordBits()"
        )
Library_item = hiCreateMenuItem(
        ?name 'Library_item
        ?itemText "Working Library"
        ?callback "mainLibrary()"
        )
TecLibrary_item = hiCreateMenuItem(
        ?name 'TecLibrary_item
        ?itemText "Technology Library"
        ?callback "mainTecLibrary()"
        )
Workpath_item = hiCreateMenuItem(
        ?name 'Workpath_item
        ?itemText "Working Path"
        ?callback "mainWorkpath()"
        )
Ports_item = hiCreateMenuItem(
        ?name 'Ports_item
        ?itemText "Ports"
        ?callback "mainPorts()"
        )
other_item = hiCreateMenuItem(
        ?name 'other_item
        ?itemText "other..."
        ?callback "mainOther()"
        )
OLayout_item = hiCreateMenuItem(
        ?name 'OLayout_item
```

```
        ?itemText "Layout"
        ?callback "mainOLayout()"

        )
OAbstract_item = hiCreateMenuItem(
        ?name 'OAbstract
        ?itemText "Abstract(SE)"
        ?callback "mainOAbstract()"
        )
OSymbol_item = hiCreateMenuItem(
        ?name 'OSymbol_item
        ?itemText "Symbol"
        ?callback "mainOSymbol()"
        )
OVerilog_item = hiCreateMenuItem(
        ?name 'OVerilog_item
        ?itemText "Verilog"
        ?callback "mainOVerilog()"
        )
OTestVec_item = hiCreateMenuItem(
        ?name 'OTestVec_item
        ?itemText "Test Vector"
        ?callback "mainOTestVec()"
        )
OVhdl_item = hiCreateMenuItem(
        ?name 'OVhdl_item
        ?itemText "Vhdl"
        ?callback "mainOVhdl()"
        )
hiCreatePulldownMenu(
    'mainFileMenu
```

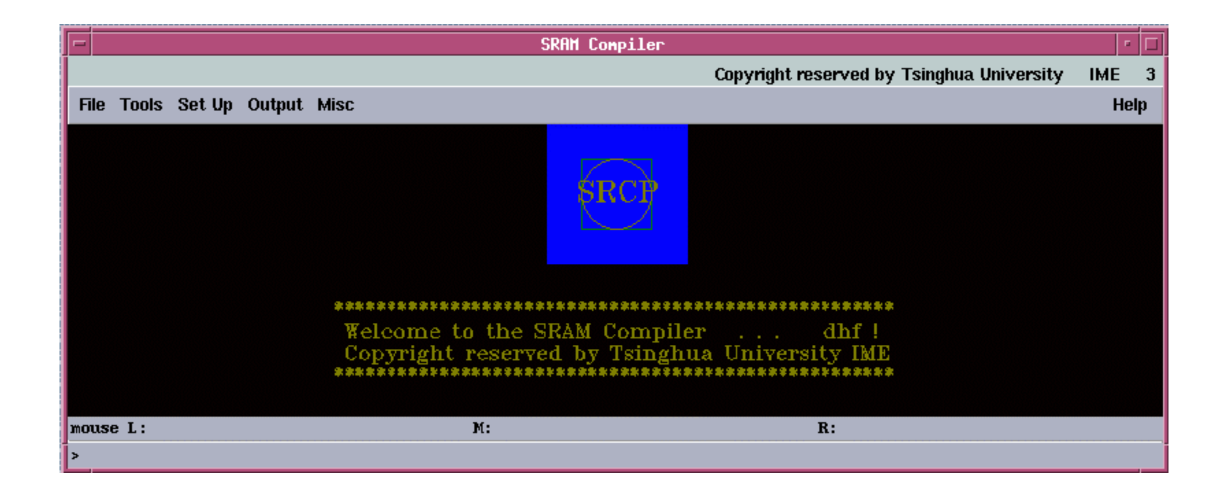
```
    "File"
    list(Exit_item)
;   list(SaveState_item LoadState_item  Exit_item)
    )
hiCreatePulldownMenu(
        'mainSetUpMenu
        "Set Up"
        list(  ModuleName_item  Words_item    Ports_item  Library_item
TecLibrary_item Workpath_item other_item)
        )
hiCreatePulldownMenu(
        'mainOutputMenu
        "Output"
        list(  OVhdl_item  OVerilog_item  OSymbol_item  OLayout_item
OAbstract_item OTestVec_item)

        )
hiCreatePulldownMenu(
        'mainMiscMenu
        "Misc"
        list( Report_item About_item  )
        )
/* First, make the DisplayList. */
dl = dlMakeDisplayList( )
/* Now make a pen table for it. */
penTable = dlMakePenTable(5)
/* Assign the penTable to the display list. */
dlSetPenTable( dl penTable)
/* Define a couple of colors. */
colorIndex = hiMatchColor( nameToColor( "blue"))
dlSetPenColor( 1 colorIndex penTable)
```

```
/* Set this pen "filled." */
dlSetPenFillStyle( 1 "SolidFill" penTable)
/* Put the objects in. */
dlAddBox( dl 1 30:30 50:50)
dlAddBox( dl 2 35:35 45:45)
dlAddCircle( dl 3 40:40 5)
sprintf( welSpring "
************************************************ \n
Welcome to the SRAM Compiler . . . %s ! \n Copyright
reserved by Tsinghua University IME \n
************************************************ \n"
getShellEnvVar("USER"))
dlAddStrokeText( dl 3 40:25 welSpring "upperCenter" "roman" 2 "0" )
dlAddStrokeText( dl 3 40:43 "SRCP" "upperCenter" "roman" 3 "0" )
;dlAddRasterText( dl 4 50:50 welSpring "roman" )
/* Make a window. Put the DisplayList into the window */
/* and set the window's icon. */

w = hiOpenWindow(
                ?bBox list(50:50 850:350)
        ?type "graphics"
;       ?appType "SRAM Compiler"
        ?menus list('mainFileMenu icfbToolsMenu 'mainSetUpMenu
'mainOutputMenu 'mainMiscMenu)
;       ?menus list('mainFileMenu 'mainSetUpMenu 'mainOutputMenu
'mainMiscMenu)
        ?labels list( " Copyright reserved by Tsinghua University " " IME
" )
        ?help "SRAMCompilerHelp "
        ?scroll nil
;       ?closeProc "mainCloseProc"
```

```
)
hiSetWindowName(
        w
        "SRAM Compiler"
        )
hiSetIconName(
        w
        "SRAM Compiler"
        )
dlAttachDlistToWindow( dl w)
/* Convert the dlist to an icon. */
icon = dlDlistToIcon( dl 50 50)
hiSetWindowIcon( w icon)
/* Save it to a file. */
dlSaveDlist( dl "save.dlist" "newDl")
/* Try resizing the window. The dlist will fit inside it. */
/* Iconify the window. */
```



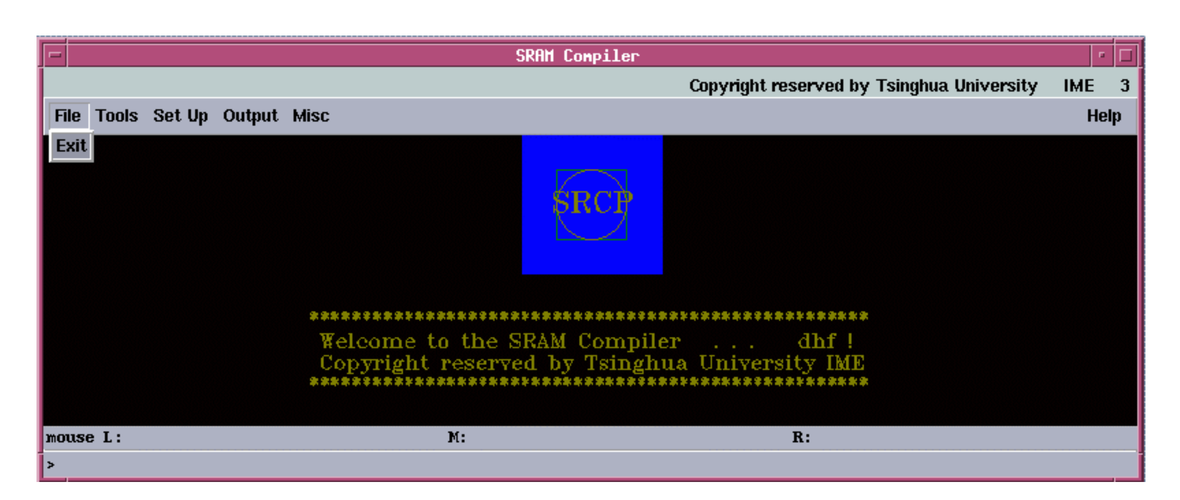


图 7－2 SRAM 编译器的 File 菜单

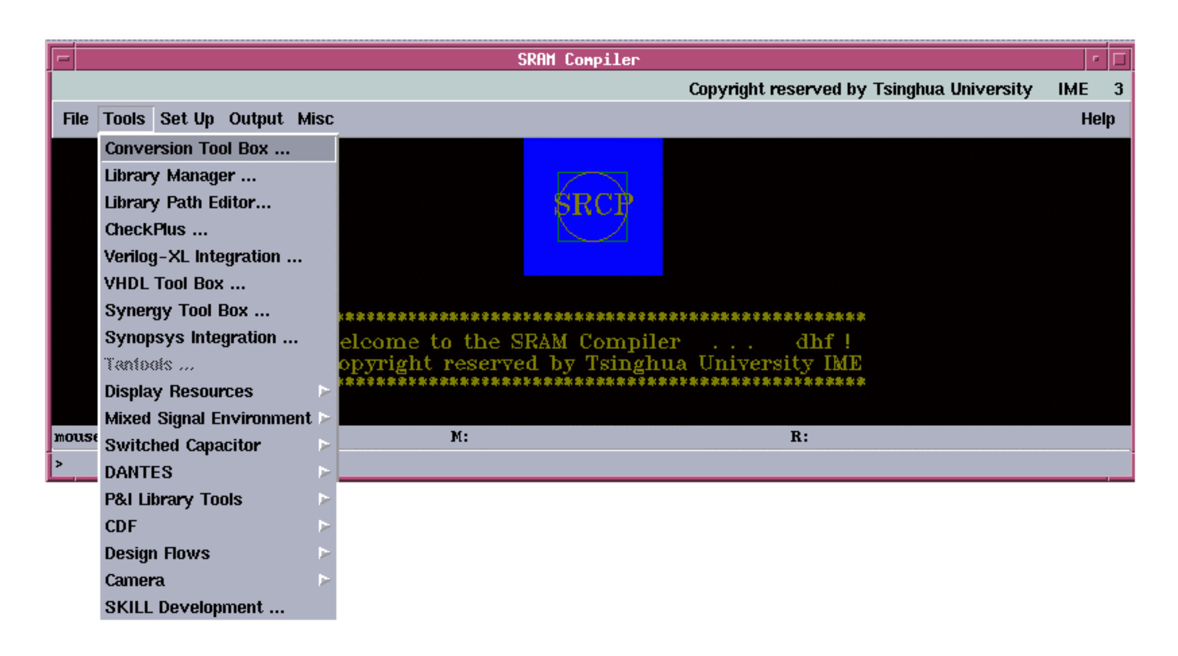


图 7－3 SRAM 编译器的 Tools 菜单

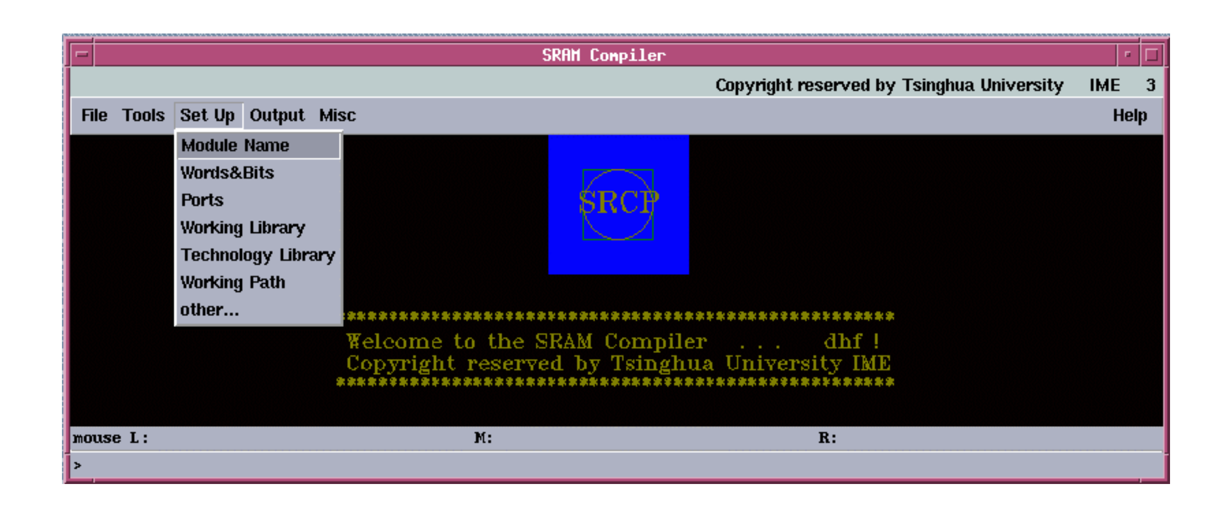


图 7－4 SRAM 编译器的 Set Up 菜单

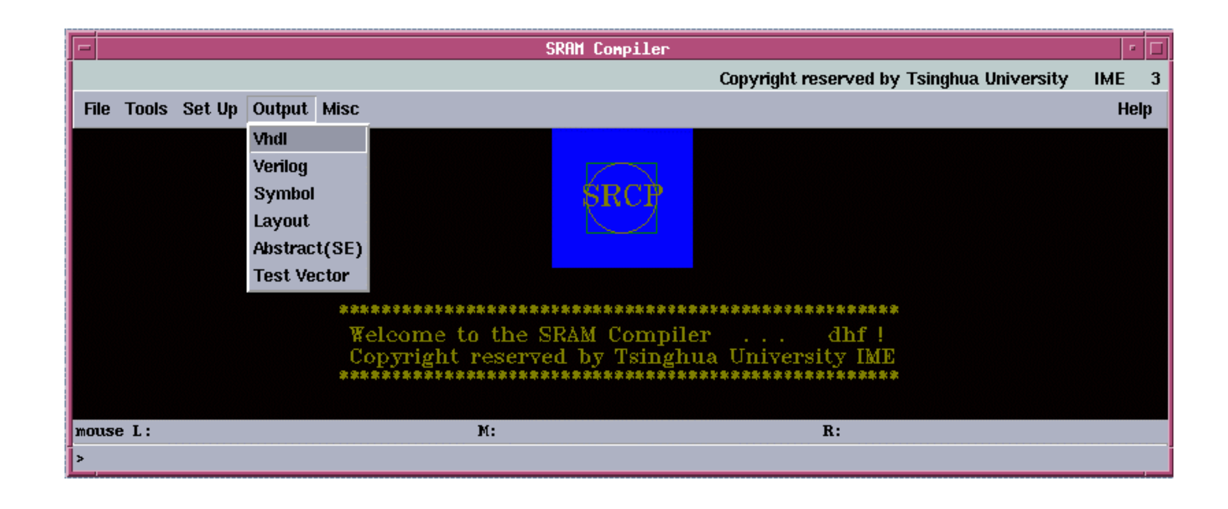


图 7－5 SRAM 编译器的 Output 菜单

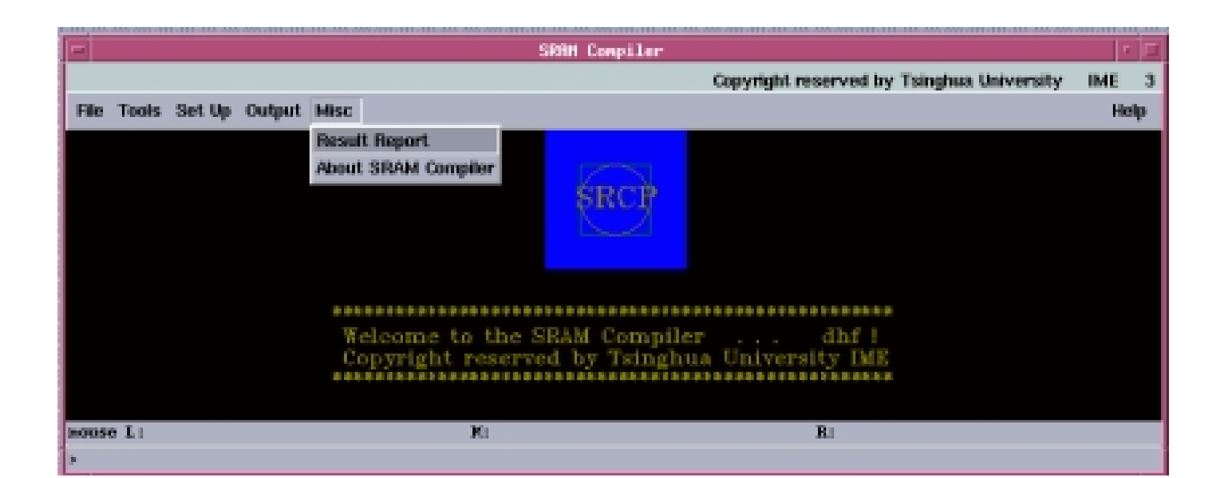


图 7－6SRAM 编译器的 Misc 菜单

附录 1 技术文件及显示文件示例

Technology File and Display Resource File Examples

This appendix contains:

Technology File Example

Display Resource File Example

Technology File Example

The following is a example of an ASCII technology file.

```
; Generated on Nov 14 07:50:20 1996
;     with layoutPlus version 4.4.1 Wed Nov 13 22:29:27 PST 1996
(cds3003)

;********************************
; Controls DEFINITION
;********************************
controls(
   techParams(theta 2.0)
)
physicalRules(
     techDefineSpacingRule(
          (minWidth metal1 techGetParam("theta") * 2))
)

;********************************
; LAYER DEFINITION
;********************************
layerDefinitions(
```

```
techPurposes(
;( PurposeName            Purpose#   Abbreviation )
;( -----------            --------   ------------ )
;User-Defined Purposes:
;System-Reserved Purposes:
 ( drawing1              241       dr1         )
 ( pin                  251       pin         )
) ;techPurposes
techLayers(
;( LayerName              Layer#    Abbreviation )
;( ---------              ------    ------------ )
;User-Defined Layers:
 ( ndiff                 1         ndiff       )
 ( pwell                 6         pwell       )
;System-Reserved Layers:
 ( Unrouted              200       unRoute     )
 ( Row                  201       Row         )
) ;techLayers

techLayerPurposePriorities(
;layers are ordered from lowest to highest priority
;( LayerName              Purpose    )
;( ---------              -------    )
 ( nimplant              drawing    )
 ( nwell                 net       )
 ( nwell                 drawing    )
) ;techLayerPurposePriorities

techDisplays(
;( LayerName    Purpose     Packet         Vis Sel Con2ChgLy
```

```
DrgEnbl Valid )
 ;( ---------    -------      ------         --- --- ---------
------- ----- )
  ( nimplant    drawing     cyan           t t t t t )
  ( nwell       net         yellow         t t nil t nil )
  ( nwell       drawing     yellow         t t t t t )
 ) ;techDisplays

techLayerProperties(
;( PropName             Layer1 [ Layer2 ]          PropValue )
 ( defaultWidth          ndiff                       1.000000 )
 ( defaultWidth          pdiff                       1.000000 )
)

) ;layerDefinitions

;*********************************
; LAYER RULES
;*********************************
layerRules(

 streamLayers(
 ;( layer       streamNumber    dataType       translate )
 ;( -----       ------------    --------       --------- )
  ( ("ptap" "drawing") 0        0              nil       )
  ( ("ptap" "net")     0        0              nil       )
  ( ndiff              0        0              nil       )
 ) ;streamLayers

 viaLayers(
 ;( layer1     viaLayer       layer2    )
```

```
 ;( ------     --------       ------    )
  ( poly1      cont            metal1    )
  ( metal1     via             metal2    )
 ) ;viaLayers

 equivalentLayers(
 ;( list of layers )
 ;( -------------- )
  ( vapox        metal3    )
 ) ;equivalentLayers

) ;layerRules

;********************************
; DEVICES
;********************************
devices(
tcCreateCDSDeviceClass()
symEnhancementDevice(
; (name sdLayer sdPurpose gateLayer gatePurpose w l sdExt gateExt
;  legalRegion)

  (PTR pdiff drawing poly1 drawing 3 1 1.5 1
  (outside pwell drawing))
)
;
; no syDepletion devices
;

symContactDevice(
; (name viaLayer viaPurpose layer1 purpose1 layer2 purpose2
```

```
;  w l (row column xPitch yPitch xBias yBias) encByLayer1 encByLayer2 legalRegion)

  (M1_P cont drawing metal1 drawing pdiff drawing
  1 1 (1 1 1.5 1.5 center center) 0.5 0.5 _NA_)
)

tfcDefineDeviceProp(
; (viewName       deviceName       propName        propValue)
  (symbolic              PTAP                          function "substrateContact")
  (symbolic              NTAP                          function "substrateContact")
)

symPinDevice(
; (name maskable layer1 purpose1 w1 layer2 purpose2 w2 legalRegion)
  (bigM1_pin t metal1 drawing 3 _NA_ _NA_ _NA_ _NA_)
  (pdiff_T nil pdiff drawing 1 _NA_ _NA_ _NA_ (outside pwell drawing))
  (ndiff_T nil ndiff drawing 1 _NA_ _NA_ _NA_ (inside pwell drawing))
)
;
; no ruleContact devices
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Opus Symbolic Device Class Definition
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; no other device classes
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Opus Symbolic Device Declaration
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; no other devices
;
) ;devices

;********************************
; PHYSICAL RULES
;********************************
physicalRules(

 orderedSpacingRules(
 ;( rule              layer1   layer2      value   )
 ;( ----              ------      ------      -----   )
  ( minEnclosure         "cellBoundary"  "nwell"   0.1 )
  ( minEnclosure         "ndiff"   "cont"0.5 )
 ) ;orderedSpacingRules

 spacingRules(
 ;( rule              layer1      layer2      value   )
 ;( ----              ------      ------      -----   )
  ( minSpacing           "ndiff"     1.0 )
  ( minSpacing           "pdiff"     1.0 )
 ) ;spacingRules

 mfgGridResolution(
     ( 0.100000 )
 ) ;mfgGridResolution
```

```
) ;physicalRules

;*******************************
; COMPACTOR RULES
;*******************************
compactorRules(
compactorLayers(
 ;( layer                    usage     )
 ;( -----                    -----     )
  ( ndiff                   "diffusion" )
  ( pdiff                   "diffusion" )
 ) ;compactorLayers
symWires(
;(name  layer  [(impLayer  impSpacing)]  [(default  min  max)]
[(legalRegion regionLayer)] [WLM])
   ( metal2    ("metal2" "drawing")   nil  (0.6 nil nil)   )
  ( metal1    ("metal1" "drawing")   nil  (0.6 nil nil)   )
 ) ;symWires

) ;compactorRules

;*******************************
; LAS RULES
;*******************************
lasRules(

 lasLayers(
 ;( layer                    usage      )
 ;( -----                    -----      )
  ( ndiff                   "ndiffLayer" )
  ( pdiff                   "pdiffLayer" )
```

```
  ( ("pwell" "drawing")      "pwellLayer" )
 ) ;lasLayers

 lasDevices(
 ;( cellview                Las name   )
 ;( --------                --------   )
 ) ;lasDevices

) ;lasRules

;********************************
; LE RULES
;********************************
leRules(

 leLswLayers(
 ;( layer             purpose        )
 ;( -----             -------        )
  ( metal1Res          drawing        )
  ( text              drawing        )
 ) ;leLswLayers
) ;leRules
;********************************
; P&R RULES
;********************************
prRules(

 prRoutingLayers(
 ;( layer                  preferredDirection  )
 ;( -----                  ------------------  )
  ( poly1                  "halfRoute" )
```

```
 ( metal1                   "horizontal" )
) ;prRoutingLayers

prMastersliceLayers(
;( layers : listed in order of lowest (closest to substrate) to
highest )
;( -------------------------------------------------------------
----------- )
 ( ndiff       pdiff      )
) ;prMastersliceLayers

) ;prRules
```

Display Resource File Example

The following is an example of a display resource (display.drf) file.

```
drDefineDisplay(
;( DisplayName   #Colors    #Stipple   #LineStyles  )
 ( display      52         32         32           )
 ( psb          32         32         32           )
 ( hp8          32         32         32           )
 ( versatecb    32         32         32           )
)
drDefineColor(
;( DisplayName   ColorsName   Red      Green    Blue     )
 ( display      white       255      255      255      )
 ( display      yellow      255      255      0        )
 ( display      silver      217      230      255      )
)
drDefineStipple(
;( DisplayName   StippleName  Bitmap     )
```

```
 ( display      dots          ( ( 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  ) ) )
)
drDefineLineStyle(
;( DisplayName   LineStyle    Size      Pattern   )
 ( display      none        1          () )
 ( display      solid       1          (1 1 1 ) )
)
drDefinePacket(
;(  DisplayName    PacketName      Stipple     LineStyle   Fill
      Outline    )
 (  display         bluevZigZag     vZigZag     solid        blue
    blue       )
 ( display       whiteXBlink    x          solid      whiteBlink
 whiteBlink)
 )
)
```

```
drDefineColor(
;( DisplayName   ColorsName   Red      Green    Blue     )
 ( psb          white       255      255      255      )
 ( psb          black       0        0        0        )
)
drDefineStipple(
;( DisplayName   StippleName  Bitmap     )
 ( psb          dots        ( ( 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  )
                              ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  ) ) )
)
drDefineLineStyle(
;( DisplayName   LineStyle    Size       Pattern   )
 ( psb          none        1          () )
 ( psb          solid       1          (1 1 1 ) )
)
drDefinePacket(
;( DisplayName    PacketName       Stipple     LineStyle   Fill
```

```
Outline    )
 ( psb            blackVcc2S      vcc2S      solid      black
black     )
 ( psb            blackBackSlash  backSlash  solid      black
black     )
 )
)
drDefineStipple(
;( DisplayName   StippleName  Bitmap     )
)
drDefineLineStyle(
;( DisplayName   LineStyle    Size      Pattern   )
 ( hp8          none        1         () )
)
drDefineStipple(
;( DisplayName   StippleName  Bitmap     )
)
drDefineLineStyle(
;( DisplayName   LineStyle    Size      Pattern   )
 ( versatecb    none        1         () )
)

drDefinePacketAlias( "psb" "metal1"    "blackChecker")
drDefinePacketAlias( "psb" "metal2"    "blackChecker")
drDefinePacketAlias( "psb" "net"       "blackChecker")
drDefinePacketAlias( "psb" "net2"      "blackChecker")
```

附录 2 Verilog-XL 实例文件

1．Test_memory.v

```
`timescale 1ns/10ps
`include "SRAM256X8.v"

module test_system;

reg  [7:0] SRAM_ADDR_REG;
reg  [7:0] SRAM_VALUE_REG;
reg  SRAM_OEB_REG
reg  SRAM_WEB_REG
reg  clock

wire  [7:0] SRAM_ADDR;
wire  [7:0] SRAM_VALUE;
wire  SRAM_OEB
wire  SRAM_WEB
wire  SRAM_OUT

assign SRAM_ADDR = SRAM_ADDR_REG
assign SRAM_VALUE = SRAM_VALUE_REG
assign SRAM_WEB   = SRAM_WEB_REG
assign SRAM_OEB   = SRAM_OEB_REG

u_SRAM256X8  SRAM256X8(clk,  SRAM_OEB,  SRAM_WEB,  SRAM_ADDR,  SRAM_VALUE,
SRAM_OUT)
```

```
initial begin
        SRAM_ADDR_REG = 8'd0;
        SRAM_VALUE_REG = 8'hff;
        SRAM_WEB_REG = 0;
        SRAM_OEB_REB = 1;

      #1590050 ;
        for(i=0;i<1024;i=i+1) begin
           SRAM_ADDR_REG = SRAM_ADDR_REG + 1;
              if(SRAM_ADDR_REG > 8'h00)
               SRAM_reg_value = SRAM_reg_value - 1;
              else SRAM_reg_value = 8'hFF;
           #100 ;
        end
        #1000 k =0 ;
               SRAM_WEB_REG = 1
              SRAM_OEB_REG =0;
       #100000
       $stop;
end

always #50 clk = ~clk;
endmodule
```

2. SRAM256X8.v

```
// Technology:   TCB773 (TSMC Design Rule)
// Created by:   -f, TSMC ASIC
// Created Date: Wed Nov 19 19:36:14 CST 1997
```

```
`celldefine
`suppress_faults
`timescale 1 ns / 10 ps
`include "ram_sy1s_8052.v"
module SRAM256X8 (CEB ,OEB ,WEB ,A ,IN ,OUT);

  parameter numAddr   = 8 ;
  parameter numOut = 8 ;
  parameter wordDepth = 256 ;

  input [7:0] A;
  input CEB;
  input WEB;
  input OEB;
  input [7:0] IN;
  output [7:0] OUT;
  reg notifier_CEB_WEB ;
  reg notifier_CEB_A ;
  reg notifier_CEB_IN ;
  reg notifier_CEB ;
  tri1 check_CEB_WEB ;
  tri1 check_CEB_A ;
  tri1 check_CEB_IN ;
  tri1 check_CEB ;


  wire [7:0] A_buf;
  wire CEB_buf;
  wire WEB_buf;
  wire OEB_buf;
```

```
wire [7:0] IN_buf;
wire [7:0] OUT_buf;

buf (A_buf[0] , A[0] );
buf (A_buf[1] , A[1] );
buf (A_buf[2] , A[2] );
buf (A_buf[3] , A[3] );
buf (A_buf[4] , A[4] );
buf (A_buf[5] , A[5] );
buf (A_buf[6] , A[6] );
buf (A_buf[7] , A[7] );
buf (CEB_buf , CEB );
buf (WEB_buf , WEB );
buf (OEB_buf , OEB );
buf (IN_buf[0] , IN[0] );
buf (IN_buf[1] , IN[1] );
buf (IN_buf[2] , IN[2] );
buf (IN_buf[3] , IN[3] );
buf (IN_buf[4] , IN[4] );
buf (IN_buf[5] , IN[5] );
buf (IN_buf[6] , IN[6] );
buf (IN_buf[7] , IN[7] );
pmos (OUT[0] ,OUT_buf[0] ,1'b0);
pmos (OUT[1] ,OUT_buf[1] ,1'b0);
pmos (OUT[2] ,OUT_buf[2] ,1'b0);
pmos (OUT[3] ,OUT_buf[3] ,1'b0);
pmos (OUT[4] ,OUT_buf[4] ,1'b0);
pmos (OUT[5] ,OUT_buf[5] ,1'b0);
pmos (OUT[6] ,OUT_buf[6] ,1'b0);
pmos (OUT[7] ,OUT_buf[7] ,1'b0);
```

```
ram_sy1s_8052 #(
  numAddr ,
  numOut ,
  wordDepth )
  core (
  .EX(1'b0) ,
  .A ( A_buf[7:0] ) ,
  .CEB ( CEB_buf ) ,
  .WEB ( WEB_buf ) ,
  .OEB ( OEB_buf ) ,
  .IN ( IN_buf[7:0] ) ,
  .OUT ( OUT_buf[7:0] ) ,
  .notifier_CEB_WEB ( notifier_CEB_WEB ) ,
  .notifier_CEB_A ( notifier_CEB_A ) ,
  .notifier_CEB_IN ( notifier_CEB_IN ) ,
  .notifier_CEB ( notifier_CEB ) ,
  .check_CEB_WEB ( check_CEB_WEB ) ,
  .check_CEB_A ( check_CEB_A ) ,
  .check_CEB_IN ( check_CEB_IN ) ,
  .check_CEB ( check_CEB )
  );


 specify
    specparam
      CellType = "MEMORY";
    specparam
      TristateDisable$OEB$OUT = "OEB",
      TristateEnable$OEB$OUT = "OEB";
    specparam
      F_Prop$CEB$OUT = 4.70,
```

```
  F_Ramp$CEB$OUT = 0.30,
  R_Prop$CEB$OUT = 4.70,
  R_Ramp$CEB$OUT = 0.30,
  disable$F_Prop$OEB$OUT = 1.10,
  disable$F_Ramp$OEB$OUT = 0.00,
  disable$R_Prop$OEB$OUT = 1.10,
  disable$R_Ramp$OEB$OUT = 0.00,
  enable$F_Prop$OEB$OUT = 1.30,
  enable$F_Ramp$OEB$OUT = 0.30,
  enable$R_Prop$OEB$OUT = 1.30,
  enable$R_Ramp$OEB$OUT = 0.30;
specparam
  Cap$A = 0.120,
  Cap$CEB = 0.040,
  Cap$IN = 0.020,
  Cap$OEB = 0.010,
  Cap$OUT = 0.040,
  Cap$WEB = 0.040;

specparam cell_count    = 0.0000;
specparam Transistors   = 0;
specparam Width         = 503.9995;
specparam Height        = 367.0757;
specparam Power         = 230.0000;

// pin-to-pin delays
(CEB *> OUT[0])=(4.70);
(CEB *> OUT[1])=(4.70);
(CEB *> OUT[2])=(4.70);
(CEB *> OUT[3])=(4.70);
(CEB *> OUT[4])=(4.70);
```

```
	(CEB *> OUT[5])=(4.70);
	(CEB *> OUT[6])=(4.70);
	(CEB *> OUT[7])=(4.70);
	(OEB *> OUT[0])=(1.30, 1.30, 1.10);
	(OEB *> OUT[1])=(1.30, 1.30, 1.10);
	(OEB *> OUT[2])=(1.30, 1.30, 1.10);
	(OEB *> OUT[3])=(1.30, 1.30, 1.10);
	(OEB *> OUT[4])=(1.30, 1.30, 1.10);
	(OEB *> OUT[5])=(1.30, 1.30, 1.10);
	(OEB *> OUT[6])=(1.30, 1.30, 1.10);
	(OEB *> OUT[7])=(1.30, 1.30, 1.10);

	// setup & hold, minimum pulse width timing checks
	$setup(negedge A, negedge CEB &&& check_CEB_A, 0.00,
notifier_CEB_A);
	$setup(negedge IN, negedge CEB &&& check_CEB_IN, 0.00,
notifier_CEB_IN);
	$setup(negedge WEB, negedge CEB &&& check_CEB_WEB, 0.00,
notifier_CEB_WEB);
	$setup(posedge A, negedge CEB &&& check_CEB_A, 0.00,
notifier_CEB_A);
	$setup(posedge IN, negedge CEB &&& check_CEB_IN, 0.00,
notifier_CEB_IN);
	$setup(posedge WEB, negedge CEB &&& check_CEB_WEB, 0.00,
notifier_CEB_WEB);
	$hold(negedge CEB, negedge A &&& check_CEB_A, 1.30, notifier_CEB_A);
	$hold(negedge CEB, negedge IN &&& check_CEB_IN, 4.90,
notifier_CEB_IN);
	$hold(negedge CEB, negedge WEB &&& check_CEB_WEB, 4.90,
notifier_CEB_WEB);
	$hold(negedge CEB, posedge A &&& check_CEB_A, 1.30, notifier_CEB_A);
```

```
        $hold(negedge CEB, posedge IN &&& check_CEB_IN, 4.90, notifier_CEB_IN);
        $hold(negedge CEB, posedge WEB &&& check_CEB_WEB, 4.90, notifier_CEB_WEB);
        $width(posedge CEB &&& check_CEB, 0.00, 0, notifier_CEB);
        $width(negedge CEB &&& check_CEB, 0.00, 0, notifier_CEB);
   endspecify

endmodule
`nosuppress_faults
`endcelldefine
```

3. ram_sy1s_8052

```
/************************************************************/
/*      TSMC Verilog modle of memory cell ram_sy1s v1.1          */
/*      Created by Albert Hung-Chun Li  , Date 12/18/96          */
/*      Revision for conditional timing check  , 04/02/97        */
/*      Description                                              */
/*              Synchronous one port RAM separated IO            */
/*      Ports                                                    */
/*              A       : address input buffer                   */
/*              IN      : data input port                        */
/*              OUT     : data output port                       */
/*              CEB     : chip enable                            */
/*              WEB     : write enable                           */
```

```
/*          OEB    : output enable                                          */
/*      Parameters                                                          */
/*          numAddr     : number of address lines                           */
/*          numOut      : number of bits per word                           */
/*          wordDepth : number of words                                     */
/**********************************************************/
`timescale 1ns/10ps
`celldefine
module ram_sy1s_8052(EX, A, CEB, WEB, OEB, IN, OUT,
                    notifier_CEB_WEB,       notifier_CEB_A,       notifier_CEB_IN,
notifier_CEB,
                    check_CEB_WEB, check_CEB_A, check_CEB_IN, check_CEB);
parameter numAddr     =   1,
            numOut      =   1,
            wordDepth =   2;

input CEB, WEB, OEB, EX;
input [numAddr-1:0] A;
input [numOut-1:0] IN;
output [numOut-1:0] OUT;
input   notifier_CEB_WEB, notifier_CEB_A, notifier_CEB_IN, notifier_CEB;
output   check_CEB_WEB, check_CEB_A, check_CEB_IN, check_CEB;

`protect
reg check_CEB_WEB, check_CEB_A, check_CEB_IN, check_CEB;
reg      [numOut-1:0] intBus;
reg [numOut-1:0]   memory[wordDepth-1:0];
reg wr_flag, rd_flag, lastCEB;

reg [numAddr-1:0] address;
```

```
initial begin
   check_CEB_WEB = 1'b1;
   check_CEB_A      = 1'b1;
   check_CEB_IN    = 1'b0;
   check_CEB         = 1'b1;
end

//zab add
initial
#10
if(EX)    $readmemb("extend_SRAM.dat", memory);
else $readmemb("SRAM.dat", memory);
wire[7:0]      sram0 = memory[0];
wire[7:0]      sram1 = memory[1];
wire[7:0]      sram8 = memory[8];
assign OUT = OEB ? {numOut{1'bz}} : intBus;

always @(OEB)
begin
   if ((OEB === 1'bx) || (OEB === 1'bz))
      $display("%m> OEB is unknown so output unknown %.1f",$realtime);
end

always @(WEB) begin
   if (WEB===1'b0)
        check_CEB_IN = 1'b1;
   else
        check_CEB_IN = 1'b0;
end

always @(CEB) begin
```

```
    if ((CEB === 1'bx) || (CEB === 1'bz)) begin
      $display ("%m> CEB is unknown at time %.1f", $realtime);
      address = {numAddr{1'bx}};
      intBus   = {numOut{1'bx}};
    end else if ((CEB == 1'b0) && (lastCEB == 1'b1)) begin
        address = A;
        rd_flag = 0;
        wr_flag = 0;
        if (WEB == 1'b1) intBus = {numOut{1'bx}};
        if (^address === 1'bx) begin          //------ CHECK ADDRESS
          //$display("%m> Address is unknown - cannot access memory %.1f",
$realtime);
        end else if (address >= wordDepth) begin
          $display("%m> Address is out of range - cannot access memory %.1f",
$realtime);
        end else begin : rw_cycle
          if (WEB == 1'b1)   begin              //-------- READ CYCLE START
            rd_flag = 1;
            intBus = memory[address];
          end else if (WEB == 1'b0) begin //-------- WRITE CYCLE START
            wr_flag = 1;
            memory[address] = IN;
          end else begin                        //-------- UNKNOWN CYCLE
            //$display("%m> WEB is unknown at start of output cycle %.1f",
$realtime);
            intBus = {numOut{1'bx}};
            memory[address] = {numOut{1'bx}};
          end
        end
    end
    lastCEB = CEB ;
```

```
end

task MemErr;
begin
   #0 disable rw_cycle;
   if (rd_flag) intBus = {numOut{1'bx}};
   if (wr_flag) memory[address] = {numOut{1'bx}};
   rd_flag = 0;
   wr_flag = 0;
end
endtask

always @notifier_CEB_WEB begin
   $display ("%m> WEB setup/hold time not met so WEB unknown %.1f", $realtime);
   MemErr;
end

always @notifier_CEB_A begin
   $display ("%m> Address setup/hold time not met so address unknown %.1f",
$realtime);
   MemErr;
   address = {numAddr{1'bx}};
end

always @notifier_CEB_IN begin
   $display ("%m> Data setup/hold time not met so data unknown %.1f", $realtime);
   disable rw_cycle;
   if (wr_flag) memory[address] = {numOut{1'bx}};
   wr_flag = 0;
end
```

```
always @notifier_CEB begin
   $display("%m> Clock cycle time too short - cycle terminated %.1f", $realtime);
   MemErr;
end
`endprotect

endmodule
`endcelldefine
```

4．TSMC 库文件

```
// TSMC Standard Cell Function Library Ver
// Model Type: AN2Dx , Mon Nov 24 11:08:44 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN2D1 (A1, A2, Z);
    input A1, A2;
    output Z;
    and    (Z, A1, A2);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN2Dx , Mon Nov 24 11:08:44 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module AN2D2 (A1, A2, Z);
    input A1, A2;
    output Z;
    and    (Z, A1, A2);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN2Dx , Mon Nov 24 11:08:44 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN2D3 (A1, A2, Z);
    input A1, A2;
    output Z;
    and    (Z, A1, A2);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN3Dx , Mon Nov 24 11:08:44 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module AN3D1 (A1, A2, A3, Z);
    input A1, A2, A3;
    output Z;
    and    (Z, A1, A2, A3);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN3Dx , Mon Nov 24 11:08:44 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN3D2 (A1, A2, A3, Z);
    input A1, A2, A3;
    output Z;
    and    (Z, A1, A2, A3);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN3Dx , Mon Nov 24 11:08:44 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module AN3D3 (A1, A2, A3, Z);
    input A1, A2, A3;
    output Z;
    and    (Z, A1, A2, A3);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN4Dx , Mon Nov 24 11:08:44 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN4D1 (A1, A2, A3, A4, Z);
    input A1, A2, A3, A4;
    output Z;
    and    (Z, A1, A2, A3, A4);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
```

```
// Model Type: AN4Dx , Mon Nov 24 11:08:44 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN4D2 (A1, A2, A3, A4, Z);
    input A1, A2, A3, A4;
    output Z;
    and    (Z, A1, A2, A3, A4);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN4Dx , Mon Nov 24 11:08:44 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN4D3 (A1, A2, A3, A4, Z);
    input A1, A2, A3, A4;
    output Z;
    and    (Z, A1, A2, A3, A4);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN5Dx , Mon Nov 24 11:08:45 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN5D1 (A1, A2, A3, A4, A5, Z);
    input A1, A2, A3, A4, A5;
    output Z;
    and    (Z, A1, A2, A3, A4, A5);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN5Dx , Mon Nov 24 11:08:45 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN5D2 (A1, A2, A3, A4, A5, Z);
    input A1, A2, A3, A4, A5;
    output Z;
    and    (Z, A1, A2, A3, A4, A5);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
```

```
        (A3 => Z)=(0, 0);
        (A4 => Z)=(0, 0);
        (A5 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN6Dx , Mon Nov 24 11:08:45 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN6D1 (A1, A2, A3, A4, A5, A6, Z);
    input A1, A2, A3, A4, A5, A6;
    output Z;
    and    (Z, A1, A2, A3, A4, A5, A6);
    specify
        (A1 => Z)=(0, 0);
        (A2 => Z)=(0, 0);
        (A3 => Z)=(0, 0);
        (A4 => Z)=(0, 0);
        (A5 => Z)=(0, 0);
        (A6 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN6Dx , Mon Nov 24 11:08:45 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN6D2 (A1, A2, A3, A4, A5, A6, Z);
```

```
    input A1, A2, A3, A4, A5, A6;
    output Z;
    and    (Z, A1, A2, A3, A4, A5, A6);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
       (A6 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN7Dx , Mon Nov 24 11:08:45 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN7D1 (A1, A2, A3, A4, A5, A6, A7, Z);
    input A1, A2, A3, A4, A5, A6, A7;
    output Z;
    and    (Z, A1, A2, A3, A4, A5, A6, A7);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
       (A6 => Z)=(0, 0);
       (A7 => Z)=(0, 0);
    endspecify
```

```
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN7Dx , Mon Nov 24 11:08:45 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN7D2 (A1, A2, A3, A4, A5, A6, A7, Z);
    input A1, A2, A3, A4, A5, A6, A7;
    output Z;
    and    (Z, A1, A2, A3, A4, A5, A6, A7);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
       (A6 => Z)=(0, 0);
       (A7 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN8Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN8D1 (A1, A2, A3, A4, A5, A6, A7, A8, Z);
    input A1, A2, A3, A4, A5, A6, A7, A8;
    output Z;
    and    (Z, A1, A2, A3, A4, A5, A6, A7, A8);
```

```
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
       (A6 => Z)=(0, 0);
       (A7 => Z)=(0, 0);
       (A8 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AN8Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module AN8D2 (A1, A2, A3, A4, A5, A6, A7, A8, Z);
    input A1, A2, A3, A4, A5, A6, A7, A8;
    output Z;
    and    (Z, A1, A2, A3, A4, A5, A6, A7, A8);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
       (A6 => Z)=(0, 0);
       (A7 => Z)=(0, 0);
       (A8 => Z)=(0, 0);
    endspecify
```

```
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI211Dx , Mon Nov 24 11:09:10 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI211D0 (A1, A2, B, C, ZN);

    input A1, A2, B, C;
    output ZN;

    and     (A, A1, A2);
    nor     (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI211Dx , Mon Nov 24 11:09:10 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI211D1 (A1, A2, B, C, ZN);

    input A1, A2, B, C;
```

```
    output ZN;

    and     (A, A1, A2);
    nor     (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI211Dx , Mon Nov 24 11:09:10 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI211D1H (A1, A2, B, C, ZN);

    input A1, A2, B, C;
    output ZN;

    and     (A, A1, A2);
    nor     (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
```

```
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI211Dx , Mon Nov 24 11:09:10 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI211D2 (A1, A2, B, C, ZN);

    input A1, A2, B, C;
    output ZN;

    and     (A, A1, A2);
    nor     (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI211Dx , Mon Nov 24 11:09:10 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI211D3 (A1, A2, B, C, ZN);
```

```
    input A1, A2, B, C;
    output ZN;

    and     (A, A1, A2);
    nor     (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI21Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI21D0 (A1, A2, B, ZN);

    input A1, A2, B;
    output ZN;

    and     (A, A1, A2);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
```

```
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI21Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI21D1 (A1, A2, B, ZN);

    input A1, A2, B;
    output ZN;

    and    (A, A1, A2);
    nor    (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI21Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI21D1H (A1, A2, B, ZN);

    input A1, A2, B;
```

```
    output ZN;

    and     (A, A1, A2);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI21Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI21D2 (A1, A2, B, ZN);

    input A1, A2, B;
    output ZN;

    and     (A, A1, A2);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI21Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI21D3 (A1, A2, B, ZN);

    input A1, A2, B;
    output ZN;

    and    (A, A1, A2);
    nor    (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI221Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI221D0 (A1, A2, B1, B2, C, ZN);

    input A1, A2, B1, B2, C;
    output ZN;
```

```
    and     (A, A1, A2);
    and     (B, B1, B2);
    nor     (ZN, A, B, C);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI221Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI221D1 (A1, A2, B1, B2, C, ZN);

    input A1, A2, B1, B2, C;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
    nor     (ZN, A, B, C);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (C => ZN)=(0, 0);
```

```
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI221Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI221D1H (A1, A2, B1, B2, C, ZN);

    input A1, A2, B1, B2, C;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
    nor     (ZN, A, B, C);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI221Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI221D2 (A1, A2, B1, B2, C, ZN);
```

```
    input A1, A2, B1, B2, C;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
    nor     (ZN, A, B, C);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI221Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI221D3 (A1, A2, B1, B2, C, ZN);

    input A1, A2, B1, B2, C;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
    nor     (ZN, A, B, C);
    specify
       (A1 => ZN)=(0, 0);
```

```
        (A2 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
        (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI222Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI222D0 (A1, A2, B1, B2, C1, C2, ZN);

    input A1, A2, B1, B2, C1, C2;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
    and     (C, C1, C2);
    nor     (ZN, A, B, C);

    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
        (C1 => ZN)=(0, 0);
        (C2 => ZN)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI222Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI222D1 (A1, A2, B1, B2, C1, C2, ZN);

    input A1, A2, B1, B2, C1, C2;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
    and     (C, C1, C2);
    nor     (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (C1 => ZN)=(0, 0);
       (C2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI222Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module AOI222D1H (A1, A2, B1, B2, C1, C2, ZN);

    input A1, A2, B1, B2, C1, C2;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
    and     (C, C1, C2);
    nor     (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (C1 => ZN)=(0, 0);
       (C2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI222Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI222D2 (A1, A2, B1, B2, C1, C2, ZN);

    input A1, A2, B1, B2, C1, C2;
    output ZN;

    and     (A, A1, A2);
```

```
    and     (B, B1, B2);
    and     (C, C1, C2);
    nor     (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (C1 => ZN)=(0, 0);
       (C2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI222Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI222D3 (A1, A2, B1, B2, C1, C2, ZN);

    input A1, A2, B1, B2, C1, C2;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
    and     (C, C1, C2);
    nor     (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
```

```
        (A2 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
        (C1 => ZN)=(0, 0);
        (C2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI22Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI22D0 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    and    (A, A1, A2);
    and    (B, B1, B2);
    nor    (ZN, A, B);
    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
```

```
// Model Type: AOI22Dx , Mon Nov 24 11:09:11 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI22D1 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
    nor     (ZN, A, B);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI22Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI22D1H (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
```

```
    nor     (ZN, A, B);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI22Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI22D2 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
    nor     (ZN, A, B);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: AOI22Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI22D3 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    and     (A, A1, A2);
    and     (B, B1, B2);
    nor     (ZN, A, B);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI31Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI31D0 (A1, A2, A3, B, ZN);

    input A1, A2, A3, B;
    output ZN;

    and     (A, A1, A2, A3);
```

```
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI31Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI31D1 (A1, A2, A3, B, ZN);

    input A1, A2, A3, B;
    output ZN;

    and     (A, A1, A2, A3);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: AOI31Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI31D1H (A1, A2, A3, B, ZN);

    input A1, A2, A3, B;
    output ZN;

    and    (A, A1, A2, A3);
    nor    (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI31Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI31D2 (A1, A2, A3, B, ZN);

    input A1, A2, A3, B;
    output ZN;
```

```
    and     (A, A1, A2, A3);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI31Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI31D3 (A1, A2, A3, B, ZN);

    input A1, A2, A3, B;
    output ZN;

    and     (A, A1, A2, A3);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI32Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI32D0 (A1, A2, A3, B1, B2, ZN);

    input A1, A2, A3, B1, B2;
    output ZN;

    and     (A, A1, A2, A3);
    and     (B, B1, B2);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI32Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI32D1 (A1, A2, A3, B1, B2, ZN);
```

```
    input A1, A2, A3, B1, B2;
    output ZN;

    and     (A, A1, A2, A3);
    and     (B, B1, B2);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI32Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI32D1H (A1, A2, A3, B1, B2, ZN);

    input A1, A2, A3, B1, B2;
    output ZN;

    and     (A, A1, A2, A3);
    and     (B, B1, B2);
    nor     (ZN, A, B);

    specify
```

```
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI32Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI32D2 (A1, A2, A3, B1, B2, ZN);

    input A1, A2, A3, B1, B2;
    output ZN;

    and     (A, A1, A2, A3);
    and     (B, B1, B2);
    nor     (ZN, A, B);

    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: AOI32Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI32D3 (A1, A2, A3, B1, B2, ZN);

    input A1, A2, A3, B1, B2;
    output ZN;

    and     (A, A1, A2, A3);
    and     (B, B1, B2);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI33Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI33D0 (A1, A2, A3, B1, B2, B3, ZN);

    input A1, A2, A3, B1, B2, B3;
```

```
    output ZN;

    and     (A, A1, A2, A3);
    and     (B, B1, B2, B3);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (B3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI33Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI33D1 (A1, A2, A3, B1, B2, B3, ZN);

    input A1, A2, A3, B1, B2, B3;
    output ZN;

    and     (A, A1, A2, A3);
    and     (B, B1, B2, B3);
    nor     (ZN, A, B);

    specify
```

```
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
        (B3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI33Dx , Mon Nov 24 11:09:13 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI33D1H (A1, A2, A3, B1, B2, B3, ZN);

    input A1, A2, A3, B1, B2, B3;
    output ZN;

    and    (A, A1, A2, A3);
    and    (B, B1, B2, B3);
    nor    (ZN, A, B);

    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
        (B3 => ZN)=(0, 0);
    endspecify
```

```
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI33Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
module AOI33D2 (A1, A2, A3, B1, B2, B3, ZN);

    input A1, A2, A3, B1, B2, B3;
    output ZN;

    and     (A, A1, A2, A3);
    and     (B, B1, B2, B3);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (B3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AOI33Dx , Mon Nov 24 11:09:12 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module AOI33D3 (A1, A2, A3, B1, B2, B3, ZN);

    input A1, A2, A3, B1, B2, B3;
    output ZN;

    and     (A, A1, A2, A3);
    and     (B, B1, B2, B3);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (B3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AS1Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module AS1D1 (A, B, CI, ADD, S, CO);

    input A, B, CI, ADD;
    output S, CO;

    xnor       (B1, B, ADD);
    xor     (S, A, B1, CI);
```

```
    and     (n3, A, CI);
    and     (n4, B1, CI);
    and     (n5, A, B1);
    or      (CO, n3, n4, n5);

    specify
       (A => CO)=(0, 0);
       (A => S)=(0, 0);
       (ADD => CO)=(0, 0);
       (ADD => S)=(0, 0);
       (B => CO)=(0, 0);
       (B => S)=(0, 0);
       (CI => CO)=(0, 0);
       (CI => S)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: AS1Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module AS1D2 (A, B, CI, ADD, S, CO);

    input A, B, CI, ADD;
    output S, CO;

    xnor        (B1, B, ADD);
    xor     (S, A, B1, CI);
    and     (n3, A, CI);
    and     (n4, B1, CI);
```

```
    and     (n5, A, B1);
    or      (CO, n3, n4, n5);

    specify
       (A => CO)=(0, 0);
       (A => S)=(0, 0);
       (ADD => CO)=(0, 0);
       (ADD => S)=(0, 0);
       (B => CO)=(0, 0);
       (B => S)=(0, 0);
       (CI => CO)=(0, 0);
       (CI => S)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BHDx , Mon Nov 24 11:08:38 CST 1997
`timescale 1ns / 10ps
`celldefine
module BHD1 (Z);
   inout Z;
   trireg (small) #(0.01, 0.01, 1.00E30) Z;
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUF1 (I, Z);
```

```
    input I;
    output Z;
    buf    (Z, I);
    specify
       (I => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUF2 (I, Z);
    input I;
    output Z;
    buf    (Z, I);
    specify
       (I => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUF3 (I, Z);
    input I;
    output Z;
    buf    (Z, I);
```

```
    specify
       (I => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUF4 (I, Z);
    input I;
    output Z;
    buf    (Z, I);
    specify
       (I => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUF5 (I, Z);
    input I;
    output Z;
    buf    (Z, I);
    specify
       (I => Z)=(0, 0);
    endspecify
```

```
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUF6 (I, Z);
    input I;
    output Z;
    buf    (Z, I);
    specify
       (I => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFTx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUFT1 (I, OE, Z);
    input I, OE;
    output Z;

    bufif1 (Z, I, OE);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
```

```
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
         end

      specify
         (I => Z)=(0, 0);
         (OE => Z)=(0, 0, 0, 0, 0, 0);
      endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFTx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUFT2 (I, OE, Z);
    input I, OE;
    output Z;

    bufif1 (Z, I, OE);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => Z)=(0, 0);
       (OE => Z)=(0, 0, 0, 0, 0, 0);
    endspecify
```

```
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFTx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUFT3 (I, OE, Z);
    input I, OE;
    output Z;

    bufif1 (Z, I, OE);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => Z)=(0, 0);
       (OE => Z)=(0, 0, 0, 0, 0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFTx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module BUFT4 (I, OE, Z);
    input I, OE;
    output Z;

    bufif1 (Z, I, OE);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => Z)=(0, 0);
       (OE => Z)=(0, 0, 0, 0, 0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFTx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUFT5 (I, OE, Z);
    input I, OE;
    output Z;

    bufif1 (Z, I, OE);

    always @(Z)
```

```
      begin
        if (!$test$plusargs("bus_conflict_off"))
           if ($countdrivers(Z) && (Z === 1'bx))
               $display("%t ++BUS CONFLICT++ : %m", $realtime);
      end

   specify
      (I => Z)=(0, 0);
      (OE => Z)=(0, 0, 0, 0, 0, 0);
   endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFTNx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUFTN1 (I, OEN, Z);

   input I, OEN;
   output Z;

   bufif0 (Z, I, OEN);

   always @(Z)
      begin
        if (!$test$plusargs("bus_conflict_off"))
           if ($countdrivers(Z) && (Z === 1'bx))
               $display("%t ++BUS CONFLICT++ : %m", $realtime);
      end
```

```
    specify
       (I => Z)=(0, 0);
       (OEN => Z)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFTNx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUFTN2 (I, OEN, Z);

    input I, OEN;
    output Z;

    bufif0 (Z, I, OEN);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => Z)=(0, 0);
       (OEN => Z)=(0, 0, 0, 0, 0, 0);
    endspecify
```

```
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFTNx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUFTN3 (I, OEN, Z);

    input I, OEN;
    output Z;

    bufif0 (Z, I, OEN);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => Z)=(0, 0);
       (OEN => Z)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFTNx , Mon Nov 24 11:08:40 CST 1997
```

```
`timescale 1ns / 10ps
`celldefine
module BUFTN4 (I, OEN, Z);

    input I, OEN;
    output Z;

    bufif0 (Z, I, OEN);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => Z)=(0, 0);
       (OEN => Z)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: BUFTNx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module BUFTN5 (I, OEN, Z);

    input I, OEN;
```

```
    output Z;

    bufif0 (Z, I, OEN);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => Z)=(0, 0);
       (OEN => Z)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: CNTCNx , Mon Nov 24 11:08:57 CST 1997
`timescale 1ns / 10ps
`celldefine
module CNTCN1 (CI, CP, CDN, Q, QN, CO);

    input CI, CP, CDN;
    output Q, QN, CO;

    buf          (CDN_i, CDN);
    reg notifier;
    pullup       (SDN);
```

```
    and          (CP_check,CDN_i,SDN);
    xor          (D, CI, Q_buf);
    tsmc_dff     (Q_buf, D, CP, CDN_i, SDN, notifier);
    buf          (Q, Q_buf);
    not          (QN, Q_buf);
    and          (CO, Q_buf, CI);

    specify
       (CDN => CO)=(0, 0);
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CI => CO)=(0, 0);
       (CP => CO)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $setup(posedge CI , posedge CP &&& CP_check, 0, notifier);
       $setup(negedge CI , posedge CP &&& CP_check, 0, notifier);
       $hold(posedge CP ,posedge  CI &&& CP_check, 0, notifier);
       $hold(posedge CP ,negedge  CI &&& CP_check, 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
```

```
// Model Type: CNTCSNx , Mon Nov 24 11:08:57 CST 1997
`timescale 1ns / 10ps
`celldefine
module CNTCSN1 (CI, CP, CDN, SDN, Q, QN, CO);

    input CI, CP, CDN, SDN;
    output Q, QN, CO;

    buf          (CDN_i, CDN);
    buf          (SDN_i, SDN);
    reg notifier;
    and          (CP_check, CDN_i, SDN_i);
    xor          (D, CI, Q_buf);
    tsmc_dff     (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf          (Q, Q_buf);
    not          (QN_buf, Q_buf);
    and          (QN, QN_buf, SDN_i);
    not     (QNN, QN);
    and          (CO, QNN, CI);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
         if (CDN_i!==1'b0) begin
         $display("%m > CDN is released at time %.2fns.", $realtime);
         end
         if (SDN_i!==1'b0) begin
         $display("%m > SDN is released at time %.2fns.", $realtime);
         end
      end
```

```
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
      if (flag == 1) begin
         $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end

    specify
       (CDN => CO)=(0, 0);
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CI => CO)=(0, 0);
       (CP => CO)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       (SDN => CO)=(0, 0);
       (SDN => Q)=(0, 0);
       (SDN => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge SDN, 0, notifier);
       $setup(posedge CI, posedge CP &&& CP_check, 0, notifier);
       $setup(negedge CI, posedge CP &&& CP_check, 0, notifier);
       $hold(posedge CP,posedge  CI &&& CP_check, 0, notifier);
       $hold(posedge CP,negedge  CI &&& CP_check, 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
```

```
       $width(posedge SDN, 0, 0, notifier);
       $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DEC2Dx , Mon Nov 24 11:08:58 CST 1997
`timescale 1ns / 10ps
`celldefine
module DEC2D1 (A0, A1, Z0N, Z1N, Z2N, Z3N);
    input A0, A1;
    output Z0N, Z1N, Z2N, Z3N;
    not     (A0N, A0);
    not     (A1N, A1);
    nand    (Z0N, A1N, A0N);
    nand    (Z1N, A1N, A0);
    nand    (Z2N, A1, A0N);
    nand    (Z3N, A1, A0);
    specify
       (A0 => Z0N)=(0, 0);
       (A0 => Z1N)=(0, 0);
       (A0 => Z2N)=(0, 0);
       (A0 => Z3N)=(0, 0);
       (A1 => Z0N)=(0, 0);
       (A1 => Z1N)=(0, 0);
       (A1 => Z2N)=(0, 0);
       (A1 => Z3N)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: DEC2Dx , Mon Nov 24 11:08:58 CST 1997
`timescale 1ns / 10ps
`celldefine
module DEC2D2 (A0, A1, Z0N, Z1N, Z2N, Z3N);
    input A0, A1;
    output Z0N, Z1N, Z2N, Z3N;
    not    (A0N, A0);
    not    (A1N, A1);
    nand   (Z0N, A1N, A0N);
    nand   (Z1N, A1N, A0);
    nand   (Z2N, A1, A0N);
    nand   (Z3N, A1, A0);
    specify
       (A0 => Z0N)=(0, 0);
       (A0 => Z1N)=(0, 0);
       (A0 => Z2N)=(0, 0);
       (A0 => Z3N)=(0, 0);
       (A1 => Z0N)=(0, 0);
       (A1 => Z1N)=(0, 0);
       (A1 => Z2N)=(0, 0);
       (A1 => Z3N)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DEC3Dx , Mon Nov 24 11:08:58 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module DEC3D0 (A0, A1, A2, Z0N, Z1N, Z2N, Z3N, Z4N, Z5N, Z6N, Z7N);
    input A0, A1, A2;
    output Z0N, Z1N, Z2N, Z3N, Z4N, Z5N, Z6N, Z7N;
    not     (A0N, A0);
    not     (A1N, A1);
    not     (A2N, A2);
    nand    (Z0N, A2N, A1N, A0N);
    nand    (Z1N, A2N, A1N, A0);
    nand    (Z2N, A2N, A1, A0N);
    nand    (Z3N, A2N, A1, A0);
    nand    (Z4N, A2, A1N, A0N);
    nand    (Z5N, A2, A1N, A0);
    nand    (Z6N, A2, A1, A0N);
    nand    (Z7N, A2, A1, A0);
    specify
       (A0 => Z0N)=(0, 0);
       (A0 => Z1N)=(0, 0);
       (A0 => Z2N)=(0, 0);
       (A0 => Z3N)=(0, 0);
       (A0 => Z4N)=(0, 0);
       (A0 => Z5N)=(0, 0);
       (A0 => Z6N)=(0, 0);
       (A0 => Z7N)=(0, 0);
       (A1 => Z0N)=(0, 0);
       (A1 => Z1N)=(0, 0);
       (A1 => Z2N)=(0, 0);
       (A1 => Z3N)=(0, 0);
       (A1 => Z4N)=(0, 0);
       (A1 => Z5N)=(0, 0);
       (A1 => Z6N)=(0, 0);
       (A1 => Z7N)=(0, 0);
```

```
        (A2 => Z0N)=(0, 0);
        (A2 => Z1N)=(0, 0);
        (A2 => Z2N)=(0, 0);
        (A2 => Z3N)=(0, 0);
        (A2 => Z4N)=(0, 0);
        (A2 => Z5N)=(0, 0);
        (A2 => Z6N)=(0, 0);
        (A2 => Z7N)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DEC3Dx , Mon Nov 24 11:08:58 CST 1997
`timescale 1ns / 10ps
`celldefine
module DEC3D1 (A0, A1, A2, Z0N, Z1N, Z2N, Z3N, Z4N, Z5N, Z6N, Z7N);
    input A0, A1, A2;
    output Z0N, Z1N, Z2N, Z3N, Z4N, Z5N, Z6N, Z7N;
    not     (A0N, A0);
    not     (A1N, A1);
    not     (A2N, A2);
    nand    (Z0N, A2N, A1N, A0N);
    nand    (Z1N, A2N, A1N, A0);
    nand    (Z2N, A2N, A1, A0N);
    nand    (Z3N, A2N, A1, A0);
    nand    (Z4N, A2, A1N, A0N);
    nand    (Z5N, A2, A1N, A0);
    nand    (Z6N, A2, A1, A0N);
    nand    (Z7N, A2, A1, A0);
    specify
```

```
        (A0 => Z0N)=(0, 0);
        (A0 => Z1N)=(0, 0);
        (A0 => Z2N)=(0, 0);
        (A0 => Z3N)=(0, 0);
        (A0 => Z4N)=(0, 0);
        (A0 => Z5N)=(0, 0);
        (A0 => Z6N)=(0, 0);
        (A0 => Z7N)=(0, 0);
        (A1 => Z0N)=(0, 0);
        (A1 => Z1N)=(0, 0);
        (A1 => Z2N)=(0, 0);
        (A1 => Z3N)=(0, 0);
        (A1 => Z4N)=(0, 0);
        (A1 => Z5N)=(0, 0);
        (A1 => Z6N)=(0, 0);
        (A1 => Z7N)=(0, 0);
        (A2 => Z0N)=(0, 0);
        (A2 => Z1N)=(0, 0);
        (A2 => Z2N)=(0, 0);
        (A2 => Z3N)=(0, 0);
        (A2 => Z4N)=(0, 0);
        (A2 => Z5N)=(0, 0);
        (A2 => Z6N)=(0, 0);
        (A2 => Z7N)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DEC3Dx , Mon Nov 24 11:08:58 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module DEC3D2 (A0, A1, A2, Z0N, Z1N, Z2N, Z3N, Z4N, Z5N, Z6N, Z7N);
    input A0, A1, A2;
    output Z0N, Z1N, Z2N, Z3N, Z4N, Z5N, Z6N, Z7N;
    not     (A0N, A0);
    not     (A1N, A1);
    not     (A2N, A2);
    nand    (Z0N, A2N, A1N, A0N);
    nand    (Z1N, A2N, A1N, A0);
    nand    (Z2N, A2N, A1, A0N);
    nand    (Z3N, A2N, A1, A0);
    nand    (Z4N, A2, A1N, A0N);
    nand    (Z5N, A2, A1N, A0);
    nand    (Z6N, A2, A1, A0N);
    nand    (Z7N, A2, A1, A0);
    specify
       (A0 => Z0N)=(0, 0);
       (A0 => Z1N)=(0, 0);
       (A0 => Z2N)=(0, 0);
       (A0 => Z3N)=(0, 0);
       (A0 => Z4N)=(0, 0);
       (A0 => Z5N)=(0, 0);
       (A0 => Z6N)=(0, 0);
       (A0 => Z7N)=(0, 0);
       (A1 => Z0N)=(0, 0);
       (A1 => Z1N)=(0, 0);
       (A1 => Z2N)=(0, 0);
       (A1 => Z3N)=(0, 0);
       (A1 => Z4N)=(0, 0);
       (A1 => Z5N)=(0, 0);
       (A1 => Z6N)=(0, 0);
```

```
        (A1 => Z7N)=(0, 0);
        (A2 => Z0N)=(0, 0);
        (A2 => Z1N)=(0, 0);
        (A2 => Z2N)=(0, 0);
        (A2 => Z3N)=(0, 0);
        (A2 => Z4N)=(0, 0);
        (A2 => Z5N)=(0, 0);
        (A2 => Z6N)=(0, 0);
        (A2 => Z7N)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DECN2Dx , Mon Nov 24 11:08:59 CST 1997
`timescale 1ns / 10ps
`celldefine
module DECN2D0 (A0, A1, EN, Z0N, Z1N, Z2N, Z3N);

    input A0, A1, EN;
    output Z0N, Z1N, Z2N, Z3N;

    not     (A0N, A0);
    not     (A1N, A1);
    not     (ENB, EN);
    nand    (Z0N, A1N, A0N, ENB);
    nand    (Z1N, A1N, A0, ENB);
    nand    (Z2N, A1, A0N, ENB);
    nand    (Z3N, A1, A0, ENB);

    specify
```

```
        (A0 => Z0N)=(0, 0);
        (A0 => Z1N)=(0, 0);
        (A0 => Z2N)=(0, 0);
        (A0 => Z3N)=(0, 0);
        (A1 => Z0N)=(0, 0);
        (A1 => Z1N)=(0, 0);
        (A1 => Z2N)=(0, 0);
        (A1 => Z3N)=(0, 0);
        (EN => Z0N)=(0, 0);
        (EN => Z1N)=(0, 0);
        (EN => Z2N)=(0, 0);
        (EN => Z3N)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DECN2Dx , Mon Nov 24 11:08:59 CST 1997
`timescale 1ns / 10ps
`celldefine
module DECN2D1 (A0, A1, EN, Z0N, Z1N, Z2N, Z3N);

    input A0, A1, EN;
    output Z0N, Z1N, Z2N, Z3N;

    not    (A0N, A0);
    not    (A1N, A1);
    not    (ENB, EN);
    nand   (Z0N, A1N, A0N, ENB);
    nand   (Z1N, A1N, A0, ENB);
    nand   (Z2N, A1, A0N, ENB);
```

```
    nand   (Z3N, A1, A0, ENB);

    specify
       (A0 => Z0N)=(0, 0);
       (A0 => Z1N)=(0, 0);
       (A0 => Z2N)=(0, 0);
       (A0 => Z3N)=(0, 0);
       (A1 => Z0N)=(0, 0);
       (A1 => Z1N)=(0, 0);
       (A1 => Z2N)=(0, 0);
       (A1 => Z3N)=(0, 0);
       (EN => Z0N)=(0, 0);
       (EN => Z1N)=(0, 0);
       (EN => Z2N)=(0, 0);
       (EN => Z3N)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DECN2Dx , Mon Nov 24 11:08:59 CST 1997
`timescale 1ns / 10ps
`celldefine
module DECN2D2 (A0, A1, EN, Z0N, Z1N, Z2N, Z3N);

    input A0, A1, EN;
    output Z0N, Z1N, Z2N, Z3N;

    not    (A0N, A0);
    not    (A1N, A1);
    not    (ENB, EN);
```

```
    nand   (Z0N, A1N, A0N, ENB);
    nand   (Z1N, A1N, A0, ENB);
    nand   (Z2N, A1, A0N, ENB);
    nand   (Z3N, A1, A0, ENB);

    specify
       (A0 => Z0N)=(0, 0);
       (A0 => Z1N)=(0, 0);
       (A0 => Z2N)=(0, 0);
       (A0 => Z3N)=(0, 0);
       (A1 => Z0N)=(0, 0);
       (A1 => Z1N)=(0, 0);
       (A1 => Z2N)=(0, 0);
       (A1 => Z3N)=(0, 0);
       (EN => Z0N)=(0, 0);
       (EN => Z1N)=(0, 0);
       (EN => Z2N)=(0, 0);
       (EN => Z3N)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DELx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module DEL2 (I, Z);
    input I;
    output Z;
    buf    (Z, I);
    specify
```

```
      (I => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DELx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module DEL3 (I, Z);
    input I;
    output Z;
    buf    (Z, I);
    specify
      (I => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DELx , Mon Nov 24 11:08:39 CST 1997
`timescale 1ns / 10ps
`celldefine
module DEL5 (I, Z);
    input I;
    output Z;
    buf    (Z, I);
    specify
      (I => Z)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFCNx , Mon Nov 24 11:09:02 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFCN1 (D, CP, CDN, Q, QN);
    input D, CP, CDN;
    output Q, QN;

    buf          (CDN_i, CDN);
    reg notifier;
    pullup (SDN);
    and          (CP_check, CDN_i, SDN);
    tsmc_dff   (Q_buf, D, CP, CDN_i, SDN, notifier);
    buf          (Q, Q_buf);
    not    (QN, Q_buf);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN , 0, notifier);
       $setup(posedge D , posedge CP &&& CP_check , 0, notifier);
       $setup(negedge D , posedge CP &&& CP_check , 0, notifier);
       $hold(posedge CP,posedge  D &&& CP_check , 0, notifier);
       $hold(posedge CP,negedge  D &&& CP_check , 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
```

```
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFCNx , Mon Nov 24 11:09:02 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFCN2 (D, CP, CDN, Q, QN);
    input D, CP, CDN;
    output Q, QN;

    buf          (CDN_i, CDN);
    reg notifier;
    pullup (SDN);
    and          (CP_check, CDN_i, SDN);
    tsmc_dff   (Q_buf, D, CP, CDN_i, SDN, notifier);
    buf          (Q, Q_buf);
    not    (QN, Q_buf);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN , 0, notifier);
       $setup(posedge D , posedge CP &&& CP_check , 0, notifier);
       $setup(negedge D , posedge CP &&& CP_check , 0, notifier);
```

```
        $hold(posedge CP,posedge  D &&& CP_check , 0, notifier);
        $hold(posedge CP,negedge  D &&& CP_check , 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFCNx , Mon Nov 24 11:09:02 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFCN3 (D, CP, CDN, Q, QN);
    input D, CP, CDN;
    output Q, QN;

    buf          (CDN_i, CDN);
    reg notifier;
    pullup (SDN);
    and          (CP_check, CDN_i, SDN);
    tsmc_dff   (Q_buf, D, CP, CDN_i, SDN, notifier);
    buf          (Q, Q_buf);
    not    (QN, Q_buf);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
```

```
        $recovery(posedge CDN,  posedge CP, 0, notifier);
        $hold(posedge CP , posedge CDN , 0, notifier);
        $setup(posedge D , posedge CP &&& CP_check , 0, notifier);
        $setup(negedge D , posedge CP &&& CP_check , 0, notifier);
        $hold(posedge CP,posedge  D &&& CP_check , 0, notifier);
        $hold(posedge CP,negedge  D &&& CP_check , 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFCNTx , Mon Nov 24 11:09:03 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFCNT1 (D, CP, CDN, OE, Q, Z);

    input D, CP, CDN, OE;
    output Q, Z;

    buf          (CDN_i, CDN);
    reg notifier;
    pullup (SDN);
    and          (CP_check, CDN_i, SDN);
    tsmc_dff   (Q_buf, D, CP, CDN_i, SDN, notifier);
    buf          (Q, Q_buf);
    bufif1 (Z, Q_buf, OE);
```

```
    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (CDN => Q)=(0, 0);
       (CDN => Z)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => Z)=(0, 0);
       (OE => Z)=(0, 0, 0, 0, 0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $setup(posedge D, posedge CP &&& CP_check, 0, notifier);
       $setup(negedge D, posedge CP &&& CP_check, 0, notifier);
       $hold(posedge CP,posedge  D &&& CP_check, 0, notifier);
       $hold(posedge CP,negedge  D &&& CP_check, 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFCNTx , Mon Nov 24 11:09:03 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module DFCNT2 (D, CP, CDN, OE, Q, Z);

    input D, CP, CDN, OE;
    output Q, Z;

    buf          (CDN_i, CDN);
    reg notifier;
    pullup (SDN);
    and          (CP_check, CDN_i, SDN);
    tsmc_dff   (Q_buf, D, CP, CDN_i, SDN, notifier);
    buf          (Q, Q_buf);
    bufif1 (Z, Q_buf, OE);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (CDN => Q)=(0, 0);
       (CDN => Z)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => Z)=(0, 0);
       (OE => Z)=(0, 0, 0, 0, 0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $setup(posedge D, posedge CP &&& CP_check, 0, notifier);
       $setup(negedge D, posedge CP &&& CP_check, 0, notifier);
       $hold(posedge CP,posedge  D &&& CP_check, 0, notifier);
```

```
       $hold(posedge CP,negedge  D &&& CP_check, 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFCSNx , Mon Nov 24 11:09:02 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFCSN1 (D, CP, CDN, SDN, Q, QN);

    input D, CP, CDN, SDN;
    output Q, QN;
    reg notifier;

    buf          (CDN_i, CDN);
    buf          (SDN_i, SDN);
    and          (CP_check, CDN_i, SDN_i);
    tsmc_dff    (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf          (Q, Q_buf);
    not          (QN_buf, Q_buf);
    and          (QN, QN_buf, SDN_i);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
```

```
        if (CDN_i!==1'b0) begin
        $display("%m > CDN is released at time %.2fns.", $realtime);
        end
        if (SDN_i!==1'b0) begin
        $display("%m > SDN is released at time %.2fns.", $realtime);
        end
      end
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
      if (flag == 1) begin
        $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       (SDN => Q)=(0, 0);
       (SDN => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge SDN, 0, notifier);
       $setup(posedge D , posedge CP &&& CP_check , 0, notifier);
       $setup(negedge D , posedge CP &&& CP_check , 0, notifier);
       $hold(posedge CP,posedge  D &&& CP_check , 0, notifier);
       $hold(posedge CP,negedge  D &&& CP_check , 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
```

```
      $width(negedge CP &&& CP_check, 0, 0, notifier);
      $width(posedge CDN, 0, 0, notifier);
      $width(negedge CDN, 0, 0, notifier);
      $width(posedge SDN, 0, 0, notifier);
      $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFCSNx , Mon Nov 24 11:09:02 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFCSN2 (D, CP, CDN, SDN, Q, QN);

    input D, CP, CDN, SDN;
    output Q, QN;
    reg notifier;

    buf          (CDN_i, CDN);
    buf          (SDN_i, SDN);
    and          (CP_check, CDN_i, SDN_i);
    tsmc_dff    (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf          (Q, Q_buf);
    not          (QN_buf, Q_buf);
    and          (QN, QN_buf, SDN_i);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
```

```
        if (CDN_i!==1'b0) begin
        $display("%m > CDN is released at time %.2fns.", $realtime);
        end
        if (SDN_i!==1'b0) begin
        $display("%m > SDN is released at time %.2fns.", $realtime);
        end
      end
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
      if (flag == 1) begin
        $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       (SDN => Q)=(0, 0);
       (SDN => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge SDN, 0, notifier);
       $setup(posedge D , posedge CP &&& CP_check , 0, notifier);
       $setup(negedge D , posedge CP &&& CP_check , 0, notifier);
       $hold(posedge CP,posedge  D &&& CP_check , 0, notifier);
       $hold(posedge CP,negedge  D &&& CP_check , 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
```

```
      $width(negedge CP &&& CP_check, 0, 0, notifier);
      $width(posedge CDN, 0, 0, notifier);
      $width(negedge CDN, 0, 0, notifier);
      $width(posedge SDN, 0, 0, notifier);
      $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFCSNx , Mon Nov 24 11:09:02 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFCSN3 (D, CP, CDN, SDN, Q, QN);

    input D, CP, CDN, SDN;
    output Q, QN;
    reg notifier;

    buf          (CDN_i, CDN);
    buf          (SDN_i, SDN);
    and          (CP_check, CDN_i, SDN_i);
    tsmc_dff    (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf          (Q, Q_buf);
    not          (QN_buf, Q_buf);
    and          (QN, QN_buf, SDN_i);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
```

```
         if (CDN_i!==1'b0) begin
         $display("%m > CDN is released at time %.2fns.", $realtime);
         end
         if (SDN_i!==1'b0) begin
         $display("%m > SDN is released at time %.2fns.", $realtime);
         end
      end
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
      if (flag == 1) begin
         $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       (SDN => Q)=(0, 0);
       (SDN => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge SDN, 0, notifier);
       $setup(posedge D , posedge CP &&& CP_check , 0, notifier);
       $setup(negedge D , posedge CP &&& CP_check , 0, notifier);
       $hold(posedge CP,posedge  D &&& CP_check , 0, notifier);
       $hold(posedge CP,negedge  D &&& CP_check , 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
```

```
        $width(negedge CP &&& CP_check, 0, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
        $width(posedge SDN, 0, 0, notifier);
        $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFCSNTx , Mon Nov 24 11:09:03 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFCSNT1 (D, CP, CDN, SDN, OE, Q, Z);

    input D, CP, CDN, SDN, OE;
    output Q, Z;

    buf          (CDN_i, CDN);
    buf          (SDN_i, SDN);
    reg notifier;
    and          (CP_check, CDN_i, SDN_i);
    tsmc_dff     (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf          (Q, Q_buf);
    bufif1       (Z, Q_buf, OE);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
         if (CDN_i!==1'b0) begin
```

```
        $display("%m > CDN is released at time %.2fns.", $realtime);
        end
        if (SDN_i!==1'b0) begin
        $display("%m > SDN is released at time %.2fns.", $realtime);
        end
      end
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
      if (flag == 1) begin
        $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end

    always @(Z)
      begin
        if (!$test$plusargs("bus_conflict_off"))
          if ($countdrivers(Z) && (Z === 1'bx))
             $display("%t ++BUS CONFLICT++ : %m", $realtime);
      end

    specify
      (CDN => Q)=(0, 0);
      (CDN => Z)=(0, 0);
      (CP => Q)=(0, 0);
      (CP => Z)=(0, 0);
      (OE => Z)=(0, 0, 0, 0, 0, 0);
      (SDN => Q)=(0, 0);
      (SDN => Z)=(0, 0);
      $recovery(posedge CDN,  posedge CP, 0, notifier);
      $hold(posedge CP , posedge CDN, 0, notifier);
```

```
      $recovery(posedge SDN,  posedge CP, 0, notifier);
      $hold(posedge CP , posedge SDN, 0, notifier);
      $setup(posedge D, posedge CP &&& CP_check, 0, notifier);
      $setup(negedge D, posedge CP &&& CP_check, 0, notifier);
      $hold(posedge CP ,posedge  D &&& CP_check, 0, notifier);
      $hold(posedge CP ,negedge  D &&& CP_check, 0, notifier);
      $width(posedge CP &&& CP_check, 0, 0, notifier);
      $width(negedge CP &&& CP_check, 0, 0, notifier);
      $width(posedge CDN, 0, 0, notifier);
      $width(negedge CDN, 0, 0, notifier);
      $width(posedge SDN, 0, 0, notifier);
      $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFCSNTx , Mon Nov 24 11:09:03 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFCSNT2 (D, CP, CDN, SDN, OE, Q, Z);

    input D, CP, CDN, SDN, OE;
    output Q, Z;

    buf          (CDN_i, CDN);
    buf          (SDN_i, SDN);
    reg notifier;
    and          (CP_check, CDN_i, SDN_i);
    tsmc_dff     (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf          (Q, Q_buf);
```

```
    bufif1       (Z, Q_buf, OE);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
         if (CDN_i!==1'b0) begin
         $display("%m > CDN is released at time %.2fns.", $realtime);
         end
         if (SDN_i!==1'b0) begin
         $display("%m > SDN is released at time %.2fns.", $realtime);
         end
      end
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
      if (flag == 1) begin
         $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (CDN => Q)=(0, 0);
       (CDN => Z)=(0, 0);
```

```
        (CP => Q)=(0, 0);
        (CP => Z)=(0, 0);
        (OE => Z)=(0, 0, 0, 0, 0, 0);
        (SDN => Q)=(0, 0);
        (SDN => Z)=(0, 0);
        $recovery(posedge CDN,  posedge CP, 0, notifier);
        $hold(posedge CP , posedge CDN, 0, notifier);
        $recovery(posedge SDN,  posedge CP, 0, notifier);
        $hold(posedge CP , posedge SDN, 0, notifier);
        $setup(posedge D, posedge CP &&& CP_check, 0, notifier);
        $setup(negedge D, posedge CP &&& CP_check, 0, notifier);
        $hold(posedge CP ,posedge  D &&& CP_check, 0, notifier);
        $hold(posedge CP ,negedge  D &&& CP_check, 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
        $width(posedge SDN, 0, 0, notifier);
        $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFFx , Mon Nov 24 11:09:01 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFF1 (D, CP, Q, QN);

    input D, CP;
    output Q, QN;
```

```
    reg notifier;

    pullup (CDN);
    pullup      (SDN);
    tsmc_dff   (Q_buf, D, CP, CDN, SDN, notifier);
    buf         (Q, Q_buf);
    not     (QN, Q_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $setup(posedge D , posedge CP, 0, notifier);
       $setup(negedge D , posedge CP, 0, notifier);
       $hold(posedge CP,posedge  D , 0, notifier);
       $hold(posedge CP,negedge  D , 0, notifier);
       $width(posedge CP, 0, 0, notifier);
       $width(negedge CP, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFFx , Mon Nov 24 11:09:01 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFF2 (D, CP, Q, QN);

    input D, CP;
    output Q, QN;
    reg notifier;
```

```
    pullup (CDN);
    pullup      (SDN);
    tsmc_dff   (Q_buf, D, CP, CDN, SDN, notifier);
    buf         (Q, Q_buf);
    not     (QN, Q_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $setup(posedge D , posedge CP, 0, notifier);
       $setup(negedge D , posedge CP, 0, notifier);
       $hold(posedge CP,posedge  D , 0, notifier);
       $hold(posedge CP,negedge  D , 0, notifier);
       $width(posedge CP, 0, 0, notifier);
       $width(negedge CP, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFFx , Mon Nov 24 11:09:01 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFF3 (D, CP, Q, QN);

    input D, CP;
    output Q, QN;
    reg notifier;

    pullup (CDN);
    pullup      (SDN);
```

```
    tsmc_dff   (Q_buf, D, CP, CDN, SDN, notifier);
    buf         (Q, Q_buf);
    not    (QN, Q_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $setup(posedge D , posedge CP, 0, notifier);
       $setup(negedge D , posedge CP, 0, notifier);
       $hold(posedge CP,posedge  D , 0, notifier);
       $hold(posedge CP,negedge  D , 0, notifier);
       $width(posedge CP, 0, 0, notifier);
       $width(negedge CP, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFSNx , Mon Nov 24 11:09:02 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFSN1 (D, CP, SDN, Q, QN);

    input D, CP, SDN;
    output Q, QN;

    buf          (SDN_i, SDN);
    reg notifier;
    pullup (CDN);
    and          (CP_check, CDN, SDN_i);
    tsmc_dff   (Q_buf, D, CP, CDN, SDN_i, notifier);
```

```
    buf          (Q, Q_buf);
    not     (QN, Q_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       (SDN => Q)=(0, 0);
       (SDN => QN)=(0, 0);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge SDN, 0, notifier);
       $setup(posedge D, posedge CP &&& CP_check, 0, notifier);
       $setup(negedge D, posedge CP &&& CP_check, 0, notifier);
       $hold(posedge CP,posedge  D &&& CP_check, 0, notifier);
       $hold(posedge CP,negedge  D &&& CP_check, 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
       $width(posedge SDN, 0, 0, notifier);
       $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFSNx , Mon Nov 24 11:09:02 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFSN2 (D, CP, SDN, Q, QN);

    input D, CP, SDN;
    output Q, QN;
```

```
    buf          (SDN_i, SDN);
    reg notifier;
    pullup (CDN);
    and          (CP_check, CDN, SDN_i);
    tsmc_dff   (Q_buf, D, CP, CDN, SDN_i, notifier);
    buf          (Q, Q_buf);
    not     (QN, Q_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       (SDN => Q)=(0, 0);
       (SDN => QN)=(0, 0);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge SDN, 0, notifier);
       $setup(posedge D, posedge CP &&& CP_check, 0, notifier);
       $setup(negedge D, posedge CP &&& CP_check, 0, notifier);
       $hold(posedge CP,posedge  D &&& CP_check, 0, notifier);
       $hold(posedge CP,negedge  D &&& CP_check, 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
       $width(posedge SDN, 0, 0, notifier);
       $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFSNx , Mon Nov 24 11:09:02 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module DFSN3 (D, CP, SDN, Q, QN);

    input D, CP, SDN;
    output Q, QN;

    buf          (SDN_i, SDN);
    reg notifier;
    pullup (CDN);
    and          (CP_check, CDN, SDN_i);
    tsmc_dff   (Q_buf, D, CP, CDN, SDN_i, notifier);
    buf          (Q, Q_buf);
    not    (QN, Q_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       (SDN => Q)=(0, 0);
       (SDN => QN)=(0, 0);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge SDN, 0, notifier);
       $setup(posedge D, posedge CP &&& CP_check, 0, notifier);
       $setup(negedge D, posedge CP &&& CP_check, 0, notifier);
       $hold(posedge CP,posedge  D &&& CP_check, 0, notifier);
       $hold(posedge CP,negedge  D &&& CP_check, 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
       $width(posedge SDN, 0, 0, notifier);
       $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: DFSNTx , Mon Nov 24 11:09:03 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFSNT1 (D, CP, SDN, OE, Q, Z);

    input D, CP, SDN, OE;
    output Q, Z;

    buf          (SDN_i, SDN);
    reg notifier;
    pullup (CDN);
    and          (CP_check, CDN, SDN_i);
    tsmc_dff   (Q_buf, D, CP, CDN, SDN_i, notifier);
    buf          (Q, Q_buf);
    bufif1 (Z, Q_buf, OE);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (CP => Q)=(0, 0);
       (CP => Z)=(0, 0);
       (OE => Z)=(0, 0, 0, 0, 0, 0);
       (SDN => Q)=(0, 0);
       (SDN => Z)=(0, 0);
```

```
        $recovery(posedge SDN,  posedge CP, 0, notifier);
        $hold(posedge CP , posedge SDN, 0, notifier);
        $setup(posedge D, posedge CP &&& CP_check, 0, notifier);
        $setup(negedge D, posedge CP &&& CP_check, 0, notifier);
        $hold(posedge CP,posedge  D &&& CP_check, 0, notifier);
        $hold(posedge CP,negedge  D &&& CP_check, 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
        $width(posedge SDN, 0, 0, notifier);
        $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFSNTx , Mon Nov 24 11:09:03 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFSNT2 (D, CP, SDN, OE, Q, Z);

    input D, CP, SDN, OE;
    output Q, Z;

    buf          (SDN_i, SDN);
    reg notifier;
    pullup (CDN);
    and          (CP_check, CDN, SDN_i);
    tsmc_dff   (Q_buf, D, CP, CDN, SDN_i, notifier);
    buf          (Q, Q_buf);
    bufif1 (Z, Q_buf, OE);
```

```
    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (CP => Q)=(0, 0);
       (CP => Z)=(0, 0);
       (OE => Z)=(0, 0, 0, 0, 0, 0);
       (SDN => Q)=(0, 0);
       (SDN => Z)=(0, 0);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge SDN, 0, notifier);
       $setup(posedge D, posedge CP &&& CP_check, 0, notifier);
       $setup(negedge D, posedge CP &&& CP_check, 0, notifier);
       $hold(posedge CP,posedge  D &&& CP_check, 0, notifier);
       $hold(posedge CP,negedge  D &&& CP_check, 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
       $width(posedge SDN, 0, 0, notifier);
       $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFTx , Mon Nov 24 11:09:03 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module DFT1 (D, CP, OE, Q, Z);

    input D, CP, OE;
    output Q, Z;

    reg notifier;
    pullup (CDN);
    pullup (SDN);
    tsmc_dff   (Q_buf, D, CP, CDN, SDN, notifier);
    buf          (Q, Q_buf);
    bufif1 (Z, Q_buf, OE);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (CP => Q)=(0, 0);
       (CP => Z)=(0, 0);
       (OE => Z)=(0, 0, 0, 0, 0, 0);
       $setup(posedge D, posedge CP , 0, notifier);
       $setup(negedge D, posedge CP , 0, notifier);
       $hold(posedge CP ,posedge  D, 0, notifier);
       $hold(posedge CP ,negedge  D, 0, notifier);
       $width(posedge CP, 0, 0, notifier);
       $width(negedge CP, 0, 0, notifier);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFTx , Mon Nov 24 11:09:03 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFT2 (D, CP, OE, Q, Z);

    input D, CP, OE;
    output Q, Z;

    reg notifier;
    pullup (CDN);
    pullup (SDN);
    tsmc_dff   (Q_buf, D, CP, CDN, SDN, notifier);
    buf          (Q, Q_buf);
    bufif1 (Z, Q_buf, OE);

    always @(Z)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(Z) && (Z === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (CP => Q)=(0, 0);
       (CP => Z)=(0, 0);
       (OE => Z)=(0, 0, 0, 0, 0, 0);
       $setup(posedge D, posedge CP , 0, notifier);
       $setup(negedge D, posedge CP , 0, notifier);
```

```
        $hold(posedge CP ,posedge  D, 0, notifier);
        $hold(posedge CP ,negedge  D, 0, notifier);
        $width(posedge CP, 0, 0, notifier);
        $width(negedge CP, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFXx , Mon Nov 24 11:09:06 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFX1 (DA, DB, SA, CP, Q, QN);

    input DA, DB, SA, CP;
    output Q, QN;

    reg notifier;
    pullup       (CDN);
    pullup       (SDN);
    not          (SB, SA);
    and          (DA_check, CDN, SDN, SA);
    and          (DB_check, CDN, SDN, SB);
    tsmc_mux     (D, DB, DA, SA);
    tsmc_dff     (Q_buf, D, CP, CDN, SDN, notifier);
    buf          (Q, Q_buf);
    not          (QN, Q_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
```

```
        $setup(posedge DA, posedge CP &&& DA_check, 0, notifier);
        $setup(negedge DA, posedge CP &&& DA_check, 0, notifier);
        $hold(posedge CP,posedge  DA &&& DA_check, 0, notifier);
        $hold(posedge CP,negedge  DA &&& DA_check, 0, notifier);
        $setup(posedge DB, posedge CP &&& DB_check, 0, notifier);
        $setup(negedge DB, posedge CP &&& DB_check, 0, notifier);
        $hold(posedge CP,posedge  DB &&& DB_check, 0, notifier);
        $hold(posedge CP,negedge  DB &&& DB_check, 0, notifier);
        $setup(posedge SA, posedge CP, 0, notifier);
        $setup(negedge SA, posedge CP, 0, notifier);
        $hold(posedge CP,posedge  SA, 0, notifier);
        $hold(posedge CP,negedge  SA, 0, notifier);
        $width(posedge CP, 0, 0, notifier);
        $width(negedge CP, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFXx , Mon Nov 24 11:09:06 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFX2 (DA, DB, SA, CP, Q, QN);

    input DA, DB, SA, CP;
    output Q, QN;

    reg notifier;
    pullup       (CDN);
    pullup       (SDN);
    not          (SB, SA);
```

```
    and           (DA_check, CDN, SDN, SA);
    and           (DB_check, CDN, SDN, SB);
    tsmc_mux      (D, DB, DA, SA);
    tsmc_dff      (Q_buf, D, CP, CDN, SDN, notifier);
    buf           (Q, Q_buf);
    not           (QN, Q_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $setup(posedge DA, posedge CP &&& DA_check, 0, notifier);
       $setup(negedge DA, posedge CP &&& DA_check, 0, notifier);
       $hold(posedge CP,posedge  DA &&& DA_check, 0, notifier);
       $hold(posedge CP,negedge  DA &&& DA_check, 0, notifier);
       $setup(posedge DB, posedge CP &&& DB_check, 0, notifier);
       $setup(negedge DB, posedge CP &&& DB_check, 0, notifier);
       $hold(posedge CP,posedge  DB &&& DB_check, 0, notifier);
       $hold(posedge CP,negedge  DB &&& DB_check, 0, notifier);
       $setup(posedge SA, posedge CP, 0, notifier);
       $setup(negedge SA, posedge CP, 0, notifier);
       $hold(posedge CP,posedge  SA, 0, notifier);
       $hold(posedge CP,negedge  SA, 0, notifier);
       $width(posedge CP, 0, 0, notifier);
       $width(negedge CP, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFXx , Mon Nov 24 11:09:06 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module DFX3 (DA, DB, SA, CP, Q, QN);

    input DA, DB, SA, CP;
    output Q, QN;

    reg notifier;
    pullup       (CDN);
    pullup       (SDN);
    not          (SB, SA);
    and          (DA_check, CDN, SDN, SA);
    and          (DB_check, CDN, SDN, SB);
    tsmc_mux     (D, DB, DA, SA);
    tsmc_dff     (Q_buf, D, CP, CDN, SDN, notifier);
    buf          (Q, Q_buf);
    not          (QN, Q_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $setup(posedge DA, posedge CP &&& DA_check, 0, notifier);
       $setup(negedge DA, posedge CP &&& DA_check, 0, notifier);
       $hold(posedge CP,posedge  DA &&& DA_check, 0, notifier);
       $hold(posedge CP,negedge  DA &&& DA_check, 0, notifier);
       $setup(posedge DB, posedge CP &&& DB_check, 0, notifier);
       $setup(negedge DB, posedge CP &&& DB_check, 0, notifier);
       $hold(posedge CP,posedge  DB &&& DB_check, 0, notifier);
       $hold(posedge CP,negedge  DB &&& DB_check, 0, notifier);
       $setup(posedge SA, posedge CP, 0, notifier);
       $setup(negedge SA, posedge CP, 0, notifier);
       $hold(posedge CP,posedge  SA, 0, notifier);
```

```
        $hold(posedge CP,negedge  SA, 0, notifier);
        $width(posedge CP, 0, 0, notifier);
        $width(negedge CP, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFXCNx , Mon Nov 24 11:09:06 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFXCN1 (DA, DB, SA, CP, CDN, Q, QN);

    input DA, DB, SA, CP, CDN;
    output Q, QN;

    buf          (CDN_i, CDN);
    reg notifier;
    pullup       (SDN);
    not          (SB, SA);
    and          (CP_check, CDN_i, SDN);
    and          (DA_check, CDN_i, SDN, SA);
    and          (DB_check, CDN_i, SDN, SB);
    tsmc_mux     (D, DB, DA, SA);
    tsmc_dff     (Q_buf, D, CP, CDN_i, SDN, notifier);
    buf          (Q, Q_buf);
    not          (QN, Q_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
```

```
        (CDN => Q)=(0, 0);
        (CDN => QN)=(0, 0);
        $recovery(posedge CDN,  posedge CP, 0, notifier);
        $hold(posedge CP , posedge CDN, 0, notifier);
        $setup(posedge DA, posedge CP &&& DA_check, 0, notifier);
        $setup(negedge DA, posedge CP &&& DA_check, 0, notifier);
        $hold(posedge CP,posedge  DA &&& DA_check, 0, notifier);
        $hold(posedge CP,negedge  DA &&& DA_check, 0, notifier);
        $setup(posedge DB, posedge CP &&& DB_check, 0, notifier);
        $setup(negedge DB, posedge CP &&& DB_check, 0, notifier);
        $hold(posedge CP,posedge DB &&& DB_check,  0, notifier);
        $hold(posedge CP,negedge DB &&& DB_check,  0, notifier);
        $setup(posedge SA, posedge CP &&& CP_check, 0, notifier);
        $setup(negedge SA, posedge CP &&& CP_check, 0, notifier);
        $hold(posedge CP,posedge  SA &&& CP_check, 0, notifier);
        $hold(posedge CP,negedge  SA &&& CP_check, 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFXCNx , Mon Nov 24 11:09:06 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFXCN2 (DA, DB, SA, CP, CDN, Q, QN);

    input DA, DB, SA, CP, CDN;
```

```
output Q, QN;

buf          (CDN_i, CDN);
reg notifier;
pullup       (SDN);
not          (SB, SA);
and          (CP_check, CDN_i, SDN);
and          (DA_check, CDN_i, SDN, SA);
and          (DB_check, CDN_i, SDN, SB);
tsmc_mux     (D, DB, DA, SA);
tsmc_dff     (Q_buf, D, CP, CDN_i, SDN, notifier);
buf          (Q, Q_buf);
not          (QN, Q_buf);

specify
   (CP => Q)=(0, 0);
   (CP => QN)=(0, 0);
   (CDN => Q)=(0, 0);
   (CDN => QN)=(0, 0);
   $recovery(posedge CDN,  posedge CP, 0, notifier);
   $hold(posedge CP , posedge CDN, 0, notifier);
   $setup(posedge DA, posedge CP &&& DA_check, 0, notifier);
   $setup(negedge DA, posedge CP &&& DA_check, 0, notifier);
   $hold(posedge CP,posedge  DA &&& DA_check, 0, notifier);
   $hold(posedge CP,negedge  DA &&& DA_check, 0, notifier);
   $setup(posedge DB, posedge CP &&& DB_check, 0, notifier);
   $setup(negedge DB, posedge CP &&& DB_check, 0, notifier);
   $hold(posedge CP,posedge DB &&& DB_check,  0, notifier);
   $hold(posedge CP,negedge DB &&& DB_check,  0, notifier);
   $setup(posedge SA, posedge CP &&& CP_check, 0, notifier);
   $setup(negedge SA, posedge CP &&& CP_check, 0, notifier);
```

```
        $hold(posedge CP,posedge  SA &&& CP_check, 0, notifier);
        $hold(posedge CP,negedge  SA &&& CP_check, 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFXCSNx , Mon Nov 24 11:09:07 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFXCSN1 (DA, DB, SA, CP, CDN, SDN, Q, QN);

    input DA, DB, SA, CP, CDN, SDN;
    output Q, QN;

    buf           (CDN_i, CDN);
    buf           (SDN_i, SDN);
    reg notifier;
    not           (SB, SA);
    and           (CP_check, CDN_i, SDN_i);
    and           (DA_check, CDN_i, SDN_i, SA);
    and           (DB_check, CDN_i, SDN_i, SB);
    tsmc_mux      (D, DB, DA, SA);
    tsmc_dff      (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf           (Q, Q_buf);
    not           (QN_buf, Q_buf);
    and           (QN, QN_buf, SDN_i);
```

```
reg flag;
always @(CDN_i or SDN_i) begin
  if (!$test$plusargs("cdn_sdn_check_off")) begin
  if (flag == 1) begin
     if (CDN_i!==1'b0) begin
     $display("%m > CDN is released at time %.2fns.", $realtime);
     end
     if (SDN_i!==1'b0) begin
     $display("%m > SDN is released at time %.2fns.", $realtime);
     end
  end
  flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
  if (flag == 1) begin
     $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
  end
end
end

specify
   (CP => Q)=(0, 0);
   (CP => QN)=(0, 0);
   (CDN => Q)=(0, 0);
   (CDN => QN)=(0, 0);
   (SDN => Q)=(0, 0);
   (SDN => QN)=(0, 0);
   $recovery(posedge CDN,  posedge CP, 0, notifier);
   $hold(posedge CP , posedge CDN, 0, notifier);
   $recovery(posedge SDN,  posedge CP, 0, notifier);
   $hold(posedge CP , posedge SDN, 0, notifier);
```

```
        $setup(posedge DA, posedge CP &&& DA_check, 0, notifier);
        $setup(negedge DA, posedge CP &&& DA_check, 0, notifier);
        $hold(posedge CP,posedge  DA &&& DA_check, 0, notifier);
        $hold(posedge CP,negedge  DA &&& DA_check, 0, notifier);
        $setup(posedge DB, posedge CP &&& DB_check, 0, notifier);
        $setup(negedge DB, posedge CP &&& DB_check, 0, notifier);
        $hold(posedge CP,posedge  DB &&& DB_check, 0, notifier);
        $hold(posedge CP,negedge  DB &&& DB_check, 0, notifier);
        $setup(posedge SA, posedge CP &&& CP_check, 0, notifier);
        $setup(negedge SA, posedge CP &&& CP_check, 0, notifier);
        $hold(posedge CP,posedge  SA &&& CP_check, 0, notifier);
        $hold(posedge CP,negedge  SA &&& CP_check, 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
        $width(posedge SDN, 0, 0, notifier);
        $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFXCSNx , Mon Nov 24 11:09:07 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFXCSN2 (DA, DB, SA, CP, CDN, SDN, Q, QN);

    input DA, DB, SA, CP, CDN, SDN;
    output Q, QN;
```

```
    buf           (CDN_i, CDN);
    buf           (SDN_i, SDN);
    reg notifier;
    not           (SB, SA);
    and           (CP_check, CDN_i, SDN_i);
    and           (DA_check, CDN_i, SDN_i, SA);
    and           (DB_check, CDN_i, SDN_i, SB);
    tsmc_mux      (D, DB, DA, SA);
    tsmc_dff      (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf           (Q, Q_buf);
    not           (QN_buf, Q_buf);
    and           (QN, QN_buf, SDN_i);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
         if (CDN_i!==1'b0) begin
         $display("%m > CDN is released at time %.2fns.", $realtime);
         end
         if (SDN_i!==1'b0) begin
         $display("%m > SDN is released at time %.2fns.", $realtime);
         end
      end
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
      if (flag == 1) begin
         $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end
```

```
specify
   (CP => Q)=(0, 0);
   (CP => QN)=(0, 0);
   (CDN => Q)=(0, 0);
   (CDN => QN)=(0, 0);
   (SDN => Q)=(0, 0);
   (SDN => QN)=(0, 0);
   $recovery(posedge CDN,  posedge CP, 0, notifier);
   $hold(posedge CP , posedge CDN, 0, notifier);
   $recovery(posedge SDN,  posedge CP, 0, notifier);
   $hold(posedge CP , posedge SDN, 0, notifier);
   $setup(posedge DA, posedge CP &&& DA_check, 0, notifier);
   $setup(negedge DA, posedge CP &&& DA_check, 0, notifier);
   $hold(posedge CP,posedge  DA &&& DA_check, 0, notifier);
   $hold(posedge CP,negedge  DA &&& DA_check, 0, notifier);
   $setup(posedge DB, posedge CP &&& DB_check, 0, notifier);
   $setup(negedge DB, posedge CP &&& DB_check, 0, notifier);
   $hold(posedge CP,posedge  DB &&& DB_check, 0, notifier);
   $hold(posedge CP,negedge  DB &&& DB_check, 0, notifier);
   $setup(posedge SA, posedge CP &&& CP_check, 0, notifier);
   $setup(negedge SA, posedge CP &&& CP_check, 0, notifier);
   $hold(posedge CP,posedge  SA &&& CP_check, 0, notifier);
   $hold(posedge CP,negedge  SA &&& CP_check, 0, notifier);
   $width(posedge CP &&& CP_check, 0, 0, notifier);
   $width(negedge CP &&& CP_check, 0, 0, notifier);
   $width(posedge CDN, 0, 0, notifier);
   $width(negedge CDN, 0, 0, notifier);
   $width(posedge SDN, 0, 0, notifier);
   $width(negedge SDN, 0, 0, notifier);
endspecify
```

```
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFXSNx , Mon Nov 24 11:09:06 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFXSN1 (DA, DB, SA, CP, SDN, Q, QN);

    input DA, DB, SA, CP, SDN;
    output Q, QN;

    buf         (SDN_i, SDN);
    reg notifier;
    pullup      (CDN);
    not         (SB, SA);
    and         (CP_check, CDN, SDN_i);
    and         (DA_check, CDN, SDN_i, SA);
    and         (DB_check, CDN, SDN_i, SB);
    tsmc_mux    (D, DB, DA, SA);
    tsmc_dff    (Q_buf, D, CP, CDN, SDN_i, notifier);
    buf         (Q, Q_buf);
    not         (QN, Q_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       (SDN => Q)=(0, 0);
       (SDN => QN)=(0, 0);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge SDN, 0, notifier);
```

```
        $setup(posedge DA, posedge CP &&& DA_check, 0, notifier);
        $setup(negedge DA, posedge CP &&& DA_check, 0, notifier);
        $hold(posedge CP,posedge  DA &&& DA_check, 0, notifier);
        $hold(posedge CP,negedge  DA &&& DA_check, 0, notifier);
        $setup(posedge DB, posedge CP &&& DB_check, 0, notifier);
        $setup(negedge DB, posedge CP &&& DB_check, 0, notifier);
        $hold(posedge CP,posedge  DB &&& DB_check, 0, notifier);
        $hold(posedge CP,negedge  DB &&& DB_check, 0, notifier);
        $setup(posedge SA, posedge CP &&& CP_check, 0, notifier);
        $setup(negedge SA, posedge CP &&& CP_check, 0, notifier);
        $hold(posedge CP,posedge  SA &&& CP_check, 0, notifier);
        $hold(posedge CP,negedge  SA &&& CP_check, 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
        $width(posedge SDN, 0, 0, notifier);
        $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: DFXSNx , Mon Nov 24 11:09:06 CST 1997
`timescale 1ns / 10ps
`celldefine
module DFXSN2 (DA, DB, SA, CP, SDN, Q, QN);

    input DA, DB, SA, CP, SDN;
    output Q, QN;

    buf          (SDN_i, SDN);
    reg notifier;
```

```
pullup      (CDN);
not         (SB, SA);
and         (CP_check, CDN, SDN_i);
and         (DA_check, CDN, SDN_i, SA);
and         (DB_check, CDN, SDN_i, SB);
tsmc_mux    (D, DB, DA, SA);
tsmc_dff    (Q_buf, D, CP, CDN, SDN_i, notifier);
buf         (Q, Q_buf);
not         (QN, Q_buf);

specify
   (CP => Q)=(0, 0);
   (CP => QN)=(0, 0);
   (SDN => Q)=(0, 0);
   (SDN => QN)=(0, 0);
   $recovery(posedge SDN,  posedge CP, 0, notifier);
   $hold(posedge CP , posedge SDN, 0, notifier);
   $setup(posedge DA, posedge CP &&& DA_check, 0, notifier);
   $setup(negedge DA, posedge CP &&& DA_check, 0, notifier);
   $hold(posedge CP,posedge  DA &&& DA_check, 0, notifier);
   $hold(posedge CP,negedge  DA &&& DA_check, 0, notifier);
   $setup(posedge DB, posedge CP &&& DB_check, 0, notifier);
   $setup(negedge DB, posedge CP &&& DB_check, 0, notifier);
   $hold(posedge CP,posedge  DB &&& DB_check, 0, notifier);
   $hold(posedge CP,negedge  DB &&& DB_check, 0, notifier);
   $setup(posedge SA, posedge CP &&& CP_check, 0, notifier);
   $setup(negedge SA, posedge CP &&& CP_check, 0, notifier);
   $hold(posedge CP,posedge  SA &&& CP_check, 0, notifier);
   $hold(posedge CP,negedge  SA &&& CP_check, 0, notifier);
   $width(posedge CP &&& CP_check, 0, 0, notifier);
   $width(negedge CP &&& CP_check, 0, 0, notifier);
```

```
        $width(posedge SDN, 0, 0, notifier);
        $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: FA1Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module FA1D1 (A, B, CI, S, CO);

    input A, B, CI;
    output S, CO;

    xor     (S, A, B, CI);
    and     (n2, A, B);
    and     (n3, A, CI);
    and     (n4, B, CI);
    or      (CO, n2, n3, n4);

    specify
       (A => CO)=(0, 0);
       (A => S)=(0, 0);
       (B => CO)=(0, 0);
       (B => S)=(0, 0);
       (CI => CO)=(0, 0);
       (CI => S)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: FA1Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module FA1D2 (A, B, CI, S, CO);

    input A, B, CI;
    output S, CO;

    xor     (S, A, B, CI);
    and     (n2, A, B);
    and     (n3, A, CI);
    and     (n4, B, CI);
    or      (CO, n2, n3, n4);

    specify
       (A => CO)=(0, 0);
       (A => S)=(0, 0);
       (B => CO)=(0, 0);
       (B => S)=(0, 0);
       (CI => CO)=(0, 0);
       (CI => S)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ICBx , Mon Nov 24 11:08:42 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module ICB0 (C, CZ);
    input C;
    output CZ;
    buf    (CZ, C);
    specify
       (C => CZ)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ICBx , Mon Nov 24 11:08:42 CST 1997
`timescale 1ns / 10ps
`celldefine
module ICB1 (C, CZ);
    input C;
    output CZ;
    buf    (CZ, C);
    specify
       (C => CZ)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ICBx , Mon Nov 24 11:08:42 CST 1997
`timescale 1ns / 10ps
`celldefine
module ICB2 (C, CZ);
    input C;
    output CZ;
```

```
    buf    (CZ, C);
    specify
       (C => CZ)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ICBSx , Mon Nov 24 11:08:43 CST 1997
`timescale 1ns / 10ps
`celldefine
module ICBS0 (C, CZ);
    input C;
    output CZ;
    buf    (CZ, C);
    specify
       (C => CZ)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ICBSx , Mon Nov 24 11:08:43 CST 1997
`timescale 1ns / 10ps
`celldefine
module ICBS1 (C, CZ);
    input C;
    output CZ;
    buf    (CZ, C);
    specify
       (C => CZ)=(0, 0);
```

```
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ICBSx , Mon Nov 24 11:08:43 CST 1997
`timescale 1ns / 10ps
`celldefine
module ICBS2 (C, CZ);
    input C;
    output CZ;
    buf    (CZ, C);
    specify
       (C => CZ)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: IND2Dx , Mon Nov 24 11:08:55 CST 1997
`timescale 1ns / 10ps
`celldefine
module IND2D0 (A1, B1, ZN);

    input A1, B1;
    output ZN;

    not    (A1N, A1);
    nand       (ZN, A1N, B1);

    specify
```

```
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: IND2Dx , Mon Nov 24 11:08:55 CST 1997
`timescale 1ns / 10ps
`celldefine
module IND2D1 (A1, B1, ZN);

    input A1, B1;
    output ZN;

    not    (A1N, A1);
    nand       (ZN, A1N, B1);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: IND2Dx , Mon Nov 24 11:08:55 CST 1997
`timescale 1ns / 10ps
`celldefine
module IND2D2 (A1, B1, ZN);
```

```
    input A1, B1;
    output ZN;

    not    (A1N, A1);
    nand       (ZN, A1N, B1);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: IND2Dx , Mon Nov 24 11:08:55 CST 1997
`timescale 1ns / 10ps
`celldefine
module IND2D3 (A1, B1, ZN);

    input A1, B1;
    output ZN;

    not    (A1N, A1);
    nand       (ZN, A1N, B1);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: IND3Dx , Mon Nov 24 11:08:55 CST 1997
`timescale 1ns / 10ps
`celldefine
module IND3D0 (A1, B1, B2, ZN);

    input A1, B1, B2;
    output ZN;

    not     (A1N, A1);
    nand        (ZN, A1N, B1, B2);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: IND3Dx , Mon Nov 24 11:08:55 CST 1997
`timescale 1ns / 10ps
`celldefine
module IND3D1 (A1, B1, B2, ZN);

    input A1, B1, B2;
    output ZN;

    not     (A1N, A1);
```

```
    nand        (ZN, A1N, B1, B2);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: IND3Dx , Mon Nov 24 11:08:55 CST 1997
`timescale 1ns / 10ps
`celldefine
module IND3D2 (A1, B1, B2, ZN);

    input A1, B1, B2;
    output ZN;

    not    (A1N, A1);
    nand        (ZN, A1N, B1, B2);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
```

```
// Model Type: IND3Dx , Mon Nov 24 11:08:55 CST 1997
`timescale 1ns / 10ps
`celldefine
module IND3D3 (A1, B1, B2, ZN);

    input A1, B1, B2;
    output ZN;

    not     (A1N, A1);
    nand       (ZN, A1N, B1, B2);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INR2Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module INR2D0 (A1, B1, ZN);

    input A1, B1;
    output ZN;

    not     (A1N, A1);
    nor     (ZN, A1N, B1);
```

```
    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INR2Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module INR2D1 (A1, B1, ZN);

    input A1, B1;
    output ZN;

    not    (A1N, A1);
    nor    (ZN, A1N, B1);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INR2Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module INR2D2 (A1, B1, ZN);
```

```
    input A1, B1;
    output ZN;

    not     (A1N, A1);
    nor     (ZN, A1N, B1);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INR2Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module INR2D3 (A1, B1, ZN);

    input A1, B1;
    output ZN;

    not     (A1N, A1);
    nor     (ZN, A1N, B1);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INR3Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module INR3D0 (A1, B1, B2, ZN);

    input A1, B1, B2;
    output ZN;

    not     (A1N, A1);
    nor     (ZN, A1N, B1, B2);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INR3Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module INR3D1 (A1, B1, B2, ZN);

    input A1, B1, B2;
    output ZN;
```

```
    not     (A1N, A1);
    nor     (ZN, A1N, B1, B2);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INR3Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module INR3D2 (A1, B1, B2, ZN);

    input A1, B1, B2;
    output ZN;

    not     (A1N, A1);
    nor     (ZN, A1N, B1, B2);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: INR3Dx , Mon Nov 24 11:08:56 CST 1997
`timescale 1ns / 10ps
`celldefine
module INR3D3 (A1, B1, B2, ZN);

    input A1, B1, B2;
    output ZN;

    not    (A1N, A1);
    nor    (ZN, A1N, B1, B2);

    specify
       (A1 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module INV0 (I, ZN);

    input I;
    output ZN;

    not    (ZN, I);
```

```
    specify
       (I => ZN)=(0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module INV1 (I, ZN);

    input I;
    output ZN;

    not    (ZN, I);

    specify
       (I => ZN)=(0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module INV2 (I, ZN);
```

```
    input I;
    output ZN;

    not    (ZN, I);

    specify
       (I => ZN)=(0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module INV3 (I, ZN);

    input I;
    output ZN;

    not    (ZN, I);

    specify
       (I => ZN)=(0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
```

```
// Model Type: INVx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module INV4 (I, ZN);

    input I;
    output ZN;

    not    (ZN, I);

    specify
       (I => ZN)=(0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module INV5 (I, ZN);

    input I;
    output ZN;

    not    (ZN, I);

    specify
       (I => ZN)=(0, 0);
    endspecify
```

```
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module INV6 (I, ZN);

    input I;
    output ZN;

    not    (ZN, I);

    specify
       (I => ZN)=(0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVTx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module INVT0 (I, OE, ZN);

    input I, OE;
    output ZN;
```

```
    notif1 (ZN, I, OE);

    always @(ZN)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(ZN) && (ZN === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => ZN)=(0, 0);
       (OE => ZN)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVTx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module INVT1 (I, OE, ZN);

    input I, OE;
    output ZN;

    notif1 (ZN, I, OE);

    always @(ZN)
       begin
         if (!$test$plusargs("bus_conflict_off"))
```

```
            if ($countdrivers(ZN) && (ZN === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => ZN)=(0, 0);
       (OE => ZN)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVTx , Mon Nov 24 11:08:40 CST 1997
`timescale 1ns / 10ps
`celldefine
module INVT2 (I, OE, ZN);

    input I, OE;
    output ZN;

    notif1 (ZN, I, OE);

    always @(ZN)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(ZN) && (ZN === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
```

```
       (I => ZN)=(0, 0);
       (OE => ZN)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVTx , Mon Nov 24 11:08:41 CST 1997
`timescale 1ns / 10ps
`celldefine
module INVT3 (I, OE, ZN);

    input I, OE;
    output ZN;

    notif1 (ZN, I, OE);

    always @(ZN)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(ZN) && (ZN === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => ZN)=(0, 0);
       (OE => ZN)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVTx , Mon Nov 24 11:08:41 CST 1997
`timescale 1ns / 10ps
`celldefine
module INVT4 (I, OE, ZN);

    input I, OE;
    output ZN;

    notif1 (ZN, I, OE);

    always @(ZN)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(ZN) && (ZN === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => ZN)=(0, 0);
       (OE => ZN)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVTx , Mon Nov 24 11:08:41 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module INVT5 (I, OE, ZN);

    input I, OE;
    output ZN;

    notif1 (ZN, I, OE);

    always @(ZN)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(ZN) && (ZN === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => ZN)=(0, 0);
       (OE => ZN)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVTNx , Mon Nov 24 11:08:41 CST 1997
`timescale 1ns / 10ps
`celldefine
module INVTN1 (I, OEN, ZN);

    input I, OEN;
    output ZN;
```

```
    notif0 (ZN, I, OEN);

    always @(ZN)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(ZN) && (ZN === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => ZN)=(0, 0);
       (OEN => ZN)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVTNx , Mon Nov 24 11:08:41 CST 1997
`timescale 1ns / 10ps
`celldefine
module INVTN2 (I, OEN, ZN);

    input I, OEN;
    output ZN;

    notif0 (ZN, I, OEN);

    always @(ZN)
       begin
```

```
      if (!$test$plusargs("bus_conflict_off"))
         if ($countdrivers(ZN) && (ZN === 1'bx))
            $display("%t ++BUS CONFLICT++ : %m", $realtime);
   end

   specify
      (I => ZN)=(0, 0);
      (OEN => ZN)=(0, 0, 0, 0, 0, 0);
   endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVTNx , Mon Nov 24 11:08:41 CST 1997
`timescale 1ns / 10ps
`celldefine
module INVTN3 (I, OEN, ZN);

   input I, OEN;
   output ZN;

   notif0 (ZN, I, OEN);

   always @(ZN)
      begin
      if (!$test$plusargs("bus_conflict_off"))
         if ($countdrivers(ZN) && (ZN === 1'bx))
            $display("%t ++BUS CONFLICT++ : %m", $realtime);
      end
```

```
    specify
       (I => ZN)=(0, 0);
       (OEN => ZN)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVTNx , Mon Nov 24 11:08:41 CST 1997
`timescale 1ns / 10ps
`celldefine
module INVTN4 (I, OEN, ZN);

    input I, OEN;
    output ZN;

    notif0 (ZN, I, OEN);

    always @(ZN)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(ZN) && (ZN === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => ZN)=(0, 0);
       (OEN => ZN)=(0, 0, 0, 0, 0, 0);
    endspecify
```

```
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: INVTNx , Mon Nov 24 11:08:41 CST 1997
`timescale 1ns / 10ps
`celldefine
module INVTN5 (I, OEN, ZN);

    input I, OEN;
    output ZN;

    notif0 (ZN, I, OEN);

    always @(ZN)
       begin
         if (!$test$plusargs("bus_conflict_off"))
            if ($countdrivers(ZN) && (ZN === 1'bx))
                $display("%t ++BUS CONFLICT++ : %m", $realtime);
       end

    specify
       (I => ZN)=(0, 0);
       (OEN => ZN)=(0, 0, 0, 0, 0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ITBx , Mon Nov 24 11:08:43 CST 1997
```

```
`timescale 1ns / 10ps
`celldefine
module ITB0 (C, CZ);

    input C;
    output CZ;

    buf    (CZ, C);

    specify
       (C => CZ)=(0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ITBx , Mon Nov 24 11:08:43 CST 1997
`timescale 1ns / 10ps
`celldefine
module ITB1 (C, CZ);

    input C;
    output CZ;

    buf    (CZ, C);

    specify
       (C => CZ)=(0, 0);
    endspecify
```

```
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ITBx , Mon Nov 24 11:08:43 CST 1997
`timescale 1ns / 10ps
`celldefine
module ITB2 (C, CZ);

    input C;
    output CZ;

    buf    (CZ, C);

    specify
       (C => CZ)=(0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ITBSx , Mon Nov 24 11:08:43 CST 1997
`timescale 1ns / 10ps
`celldefine
module ITBS0 (C, CZ);

    input C;
    output CZ;

    buf    (CZ, C);
```

```
    specify
       (C => CZ)=(0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ITBSx , Mon Nov 24 11:08:43 CST 1997
`timescale 1ns / 10ps
`celldefine
module ITBS1 (C, CZ);

    input C;
    output CZ;

    buf    (CZ, C);

    specify
       (C => CZ)=(0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ITBSx , Mon Nov 24 11:08:43 CST 1997
`timescale 1ns / 10ps
`celldefine
module ITBS2 (C, CZ);
```

```
    input C;
    output CZ;

    buf    (CZ, C);

    specify
       (C => CZ)=(0, 0);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: JKFx , Mon Nov 24 11:09:04 CST 1997
`timescale 1ns / 10ps
`celldefine
module JKF1 (J, K, CP, Q, QN);

    input J, K, CP;
    output Q, QN;
    reg notifier;

    pullup      (CDN);
    pullup      (SDN);
    nand        (KQ, K, Q_buf);
    or          (JQ, J, Q_buf);
    not         (JN, J);
    nand        (JNK, JN, K);
    and         (D, JNK, JQ, KQ);
    tsmc_dff    (Q_buf, D, CP, CDN, SDN, notifier);
```

```
    not          (QN_buf, Q_buf);
    buf          (Q, Q_buf);
    buf          (QN, QN_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $setup(posedge J, posedge CP , 0, notifier);
       $setup(negedge J, posedge CP , 0, notifier);
       $hold(posedge CP ,posedge  J, 0, notifier);
       $hold(posedge CP ,negedge  J, 0, notifier);
       $setup(posedge K, posedge CP , 0, notifier);
       $setup(negedge K, posedge CP , 0, notifier);
       $hold(posedge CP,posedge  K, 0, notifier);
       $hold(posedge CP,negedge  K, 0, notifier);
       $width(posedge CP, 0, 0, notifier);
       $width(negedge CP, 0, 0, notifier);
    endspecify

endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: JKFx , Mon Nov 24 11:09:04 CST 1997
`timescale 1ns / 10ps
`celldefine
module JKF2 (J, K, CP, Q, QN);

    input J, K, CP;
    output Q, QN;
    reg notifier;
```

```
    pullup        (CDN);
    pullup        (SDN);
    nand          (KQ, K, Q_buf);
    or            (JQ, J, Q_buf);
    not           (JN, J);
    nand          (JNK, JN, K);
    and           (D, JNK, JQ, KQ);
    tsmc_dff      (Q_buf, D, CP, CDN, SDN, notifier);
    not           (QN_buf, Q_buf);
    buf           (Q, Q_buf);
    buf           (QN, QN_buf);

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $setup(posedge J, posedge CP , 0, notifier);
       $setup(negedge J, posedge CP , 0, notifier);
       $hold(posedge CP ,posedge  J, 0, notifier);
       $hold(posedge CP ,negedge  J, 0, notifier);
       $setup(posedge K, posedge CP , 0, notifier);
       $setup(negedge K, posedge CP , 0, notifier);
       $hold(posedge CP,posedge  K, 0, notifier);
       $hold(posedge CP,negedge  K, 0, notifier);
       $width(posedge CP, 0, 0, notifier);
       $width(negedge CP, 0, 0, notifier);
    endspecify

endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: JKFCNx , Mon Nov 24 11:09:04 CST 1997
`timescale 1ns / 10ps
`celldefine
module JKFCN1 (J, K, CP, CDN, Q, QN);

    input J, K, CP, CDN;
    output Q, QN;
    reg notifier;

    buf          (CDN_i, CDN);
    pullup       (SDN);
    and     (CP_check, CDN_i, SDN);
    nand         (KQ, K, Q_buf);
    or           (JQ, J, Q_buf);
    not          (JN, J);
    nand         (JNK, JN, K);
    and          (D, JNK, JQ, KQ);
    tsmc_dff     (Q_buf, D, CP, CDN_i, SDN, notifier);
    not          (QN_buf, Q_buf);
    buf          (Q, Q_buf);
    buf          (QN, QN_buf);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $setup(posedge J, posedge CP &&& CP_check, 0, notifier);
```

```
        $setup(negedge J, posedge CP &&& CP_check, 0, notifier);
        $hold(posedge CP ,posedge  J &&& CP_check, 0, notifier);
        $hold(posedge CP ,negedge  J &&& CP_check, 0, notifier);
        $setup(posedge K, posedge CP &&& CP_check, 0, notifier);
        $setup(negedge K, posedge CP &&& CP_check, 0, notifier);
        $hold(posedge CP ,posedge  K &&& CP_check, 0, notifier);
        $hold(posedge CP ,negedge  K &&& CP_check, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: JKFCNx , Mon Nov 24 11:09:04 CST 1997
`timescale 1ns / 10ps
`celldefine
module JKFCN2 (J, K, CP, CDN, Q, QN);

    input J, K, CP, CDN;
    output Q, QN;
    reg notifier;

    buf             (CDN_i, CDN);
    pullup          (SDN);
    and        (CP_check, CDN_i, SDN);
    nand            (KQ, K, Q_buf);
    or              (JQ, J, Q_buf);
    not             (JN, J);
```

```
    nand        (JNK, JN, K);
    and         (D, JNK, JQ, KQ);
    tsmc_dff    (Q_buf, D, CP, CDN_i, SDN, notifier);
    not         (QN_buf, Q_buf);
    buf         (Q, Q_buf);
    buf         (QN, QN_buf);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $setup(posedge J, posedge CP &&& CP_check, 0, notifier);
       $setup(negedge J, posedge CP &&& CP_check, 0, notifier);
       $hold(posedge CP ,posedge  J &&& CP_check, 0, notifier);
       $hold(posedge CP ,negedge  J &&& CP_check, 0, notifier);
       $setup(posedge K, posedge CP &&& CP_check, 0, notifier);
       $setup(negedge K, posedge CP &&& CP_check, 0, notifier);
       $hold(posedge CP ,posedge  K &&& CP_check, 0, notifier);
       $hold(posedge CP ,negedge  K &&& CP_check, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
```

```
// Model Type: JKFCSNx , Mon Nov 24 11:09:04 CST 1997
`timescale 1ns / 10ps
`celldefine
module JKFCSN1 (J, K, CP, CDN, SDN, Q, QN);

    input J, K, CP, CDN, SDN;
    output Q, QN;
    reg notifier;

    buf          (CDN_i, CDN);
    buf          (SDN_i, SDN);
    and     (CP_check, CDN_i, SDN_i);
    nand         (KQ, K, Q_buf);
    or           (JQ, J, Q_buf);
    not          (JN, J);
    nand         (JNK, JN, K);
    and          (D, JNK, JQ, KQ);
    tsmc_dff     (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf          (Q, Q_buf);
    not          (QN_buf, Q_buf);
    and          (QN, QN_buf, SDN_i);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
         if (CDN_i!==1'b0) begin
         $display("%m > CDN is released at time %.2fns.", $realtime);
         end
         if (SDN_i!==1'b0) begin
         $display("%m > SDN is released at time %.2fns.", $realtime);
```

```
          end
       end
       flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
       if (flag == 1) begin
          $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
       end
     end
     end

     specify
        (CDN => Q)=(0, 0);
        (CDN => QN)=(0, 0);
        (CP => Q)=(0, 0);
        (CP => QN)=(0, 0);
        (SDN => Q)=(0, 0);
        (SDN => QN)=(0, 0);
        $recovery(posedge CDN,  posedge CP, 0, notifier);
        $recovery(posedge SDN,  posedge CP, 0, notifier);
        $hold(posedge CP , posedge SDN, 0, notifier);
        $hold(posedge CP , posedge CDN, 0, notifier);
        $setup(posedge J, posedge CP &&& CP_check, 0, notifier);
        $setup(negedge J, posedge CP &&& CP_check, 0, notifier);
        $hold(posedge CP ,posedge  J &&& CP_check, 0, notifier);
        $hold(posedge CP ,negedge  J &&& CP_check, 0, notifier);
        $setup(posedge K, posedge CP &&& CP_check, 0, notifier);
        $setup(negedge K, posedge CP &&& CP_check, 0, notifier);
        $hold(posedge CP ,posedge  K &&& CP_check, 0, notifier);
        $hold(posedge CP ,negedge  K &&& CP_check, 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
```

```
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
        $width(posedge SDN, 0, 0, notifier);
        $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: JKFCSNx , Mon Nov 24 11:09:04 CST 1997
`timescale 1ns / 10ps
`celldefine
module JKFCSN2 (J, K, CP, CDN, SDN, Q, QN);

    input J, K, CP, CDN, SDN;
    output Q, QN;
    reg notifier;

    buf          (CDN_i, CDN);
    buf          (SDN_i, SDN);
    and     (CP_check, CDN_i, SDN_i);
    nand         (KQ, K, Q_buf);
    or           (JQ, J, Q_buf);
    not          (JN, J);
    nand         (JNK, JN, K);
    and          (D, JNK, JQ, KQ);
    tsmc_dff     (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf          (Q, Q_buf);
    not          (QN_buf, Q_buf);
    and          (QN, QN_buf, SDN_i);
```

```
    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
         if (CDN_i!==1'b0) begin
         $display("%m > CDN is released at time %.2fns.", $realtime);
         end
         if (SDN_i!==1'b0) begin
         $display("%m > SDN is released at time %.2fns.", $realtime);
         end
      end
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
      if (flag == 1) begin
         $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       (SDN => Q)=(0, 0);
       (SDN => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge SDN, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $setup(posedge J, posedge CP &&& CP_check, 0, notifier);
```

```
        $setup(negedge J, posedge CP &&& CP_check, 0, notifier);
        $hold(posedge CP ,posedge  J &&& CP_check, 0, notifier);
        $hold(posedge CP ,negedge  J &&& CP_check, 0, notifier);
        $setup(posedge K, posedge CP &&& CP_check, 0, notifier);
        $setup(negedge K, posedge CP &&& CP_check, 0, notifier);
        $hold(posedge CP ,posedge  K &&& CP_check, 0, notifier);
        $hold(posedge CP ,negedge  K &&& CP_check, 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
        $width(posedge SDN, 0, 0, notifier);
        $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LHx , Mon Nov 24 11:09:07 CST 1997
`timescale 1ns / 10ps
`celldefine
module LH1 (D, E, Q, QN);

    input D, E;
    output Q, QN;
    reg notifier;

    pullup (CDN);
    pullup (SDN);
    tsmc_dla   (Q_buf, D, E, CDN, SDN, notifier);
    buf         (Q, Q_buf);
```

```
    not     (QN, Q_buf);

    specify
       (D => Q)=(0, 0);
       (D => QN)=(0, 0);
       (E => Q)=(0, 0);
       (E => QN)=(0, 0);
       $setup(posedge D, negedge E , 0, notifier);
       $setup(negedge D, negedge E , 0, notifier);
       $hold(negedge E ,posedge  D, 0, notifier);
       $hold(negedge E ,negedge  D, 0, notifier);
       $width(posedge E, 0, 0, notifier);
       $width(negedge E, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LHx , Mon Nov 24 11:09:07 CST 1997
`timescale 1ns / 10ps
`celldefine
module LH2 (D, E, Q, QN);

    input D, E;
    output Q, QN;
    reg notifier;

    pullup (CDN);
    pullup (SDN);
    tsmc_dla   (Q_buf, D, E, CDN, SDN, notifier);
    buf          (Q, Q_buf);
```

```
    not     (QN, Q_buf);

    specify
       (D => Q)=(0, 0);
       (D => QN)=(0, 0);
       (E => Q)=(0, 0);
       (E => QN)=(0, 0);
       $setup(posedge D, negedge E , 0, notifier);
       $setup(negedge D, negedge E , 0, notifier);
       $hold(negedge E ,posedge  D, 0, notifier);
       $hold(negedge E ,negedge  D, 0, notifier);
       $width(posedge E, 0, 0, notifier);
       $width(negedge E, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LHCNx , Mon Nov 24 11:09:07 CST 1997
`timescale 1ns / 10ps
`celldefine
module LHCN1 (D, E, CDN, Q, QN);

    input D, E, CDN;
    output Q, QN;

    buf           (CDN_i, CDN);
    reg notifier;
    pullup (SDN);
    and           (E_check, CDN_i, SDN);
    tsmc_dla   (Q_buf, D, E, CDN_i, SDN, notifier);
```

```
    buf          (Q, Q_buf);
    not     (QN, Q_buf);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (D => Q)=(0, 0);
       (D => QN)=(0, 0);
       (E => Q)=(0, 0);
       (E => QN)=(0, 0);
       $recovery(posedge CDN,  negedge E, 0, notifier);
       $hold(negedge E , posedge CDN, 0, notifier);
       $setup(posedge D, negedge E &&& E_check, 0, notifier);
       $setup(negedge D, negedge E &&& E_check, 0, notifier);
       $hold(negedge E ,posedge D &&& E_check, 0, notifier);
       $hold(negedge E ,negedge D &&& E_check, 0, notifier);
       $width(posedge E &&& E_check, 0, 0, notifier);
       $width(negedge E &&& E_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LHCNx , Mon Nov 24 11:09:07 CST 1997
`timescale 1ns / 10ps
`celldefine
module LHCN2 (D, E, CDN, Q, QN);

    input D, E, CDN;
```

```
    output Q, QN;

    buf          (CDN_i, CDN);
    reg notifier;
    pullup (SDN);
    and          (E_check, CDN_i, SDN);
    tsmc_dla   (Q_buf, D, E, CDN_i, SDN, notifier);
    buf          (Q, Q_buf);
    not    (QN, Q_buf);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (D => Q)=(0, 0);
       (D => QN)=(0, 0);
       (E => Q)=(0, 0);
       (E => QN)=(0, 0);
       $recovery(posedge CDN,  negedge E, 0, notifier);
       $hold(negedge E , posedge CDN, 0, notifier);
       $setup(posedge D, negedge E &&& E_check, 0, notifier);
       $setup(negedge D, negedge E &&& E_check, 0, notifier);
       $hold(negedge E ,posedge D &&& E_check, 0, notifier);
       $hold(negedge E ,negedge D &&& E_check, 0, notifier);
       $width(posedge E &&& E_check, 0, 0, notifier);
       $width(negedge E &&& E_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: LHCNx , Mon Nov 24 11:09:07 CST 1997
`timescale 1ns / 10ps
`celldefine
module LHCN3 (D, E, CDN, Q, QN);

    input D, E, CDN;
    output Q, QN;

    buf          (CDN_i, CDN);
    reg notifier;
    pullup (SDN);
    and          (E_check, CDN_i, SDN);
    tsmc_dla   (Q_buf, D, E, CDN_i, SDN, notifier);
    buf          (Q, Q_buf);
    not    (QN, Q_buf);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (D => Q)=(0, 0);
       (D => QN)=(0, 0);
       (E => Q)=(0, 0);
       (E => QN)=(0, 0);
       $recovery(posedge CDN,  negedge E, 0, notifier);
       $hold(negedge E , posedge CDN, 0, notifier);
       $setup(posedge D, negedge E &&& E_check, 0, notifier);
       $setup(negedge D, negedge E &&& E_check, 0, notifier);
       $hold(negedge E ,posedge D &&& E_check, 0, notifier);
       $hold(negedge E ,negedge D &&& E_check, 0, notifier);
       $width(posedge E &&& E_check, 0, 0, notifier);
```

```
        $width(negedge E &&& E_check, 0, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LHTx , Mon Nov 24 11:09:10 CST 1997
`timescale 1ns / 10ps
`celldefine
module LHT1 (D, E, OE, Z);

    input D, E, OE;
    output Z;
    reg notifier;

    pullup (CDN);
    pullup (SDN);
    tsmc_dla   (Q_buf, D, E, CDN, SDN, notifier);
    bufif1 (Z, Q_buf, OE);

    specify
        (D => Z)=(0, 0);
        (E => Z)=(0, 0);
        (OE => Z)=(0, 0, 0, 0, 0, 0);
        $setup(posedge D, negedge E, 0, notifier);
        $setup(negedge D, negedge E, 0, notifier);
        $hold(negedge E ,posedge  D, 0, notifier);
        $hold(negedge E ,negedge  D, 0, notifier);
        $width(posedge E, 0, 0, notifier);
```

```
      $width(negedge E, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LNx , Mon Nov 24 11:09:08 CST 1997
`timescale 1ns / 10ps
`celldefine
module LN1 (D, EN, Q, QN);

    input D, EN;
    output Q, QN;
    reg notifier;

    pullup (CDN);
    pullup (SDN);
    not    (E, EN);
    tsmc_dla   (Q_buf, D, E, CDN, SDN, notifier);
    buf          (Q, Q_buf);
    not    (QN, Q_buf);

    specify
      (D => Q)=(0, 0);
      (D => QN)=(0, 0);
      (EN => Q)=(0, 0);
      (EN => QN)=(0, 0);
      $setup(posedge D, posedge EN , 0, notifier);
      $setup(negedge D, posedge EN , 0, notifier);
      $hold(posedge EN ,posedge  D, 0, notifier);
      $hold(posedge EN ,negedge  D, 0, notifier);
```

```
        $width(posedge EN, 0, 0, notifier);
        $width(negedge EN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LNx , Mon Nov 24 11:09:09 CST 1997
`timescale 1ns / 10ps
`celldefine
module LN2 (D, EN, Q, QN);

    input D, EN;
    output Q, QN;
    reg notifier;

    pullup (CDN);
    pullup (SDN);
    not     (E, EN);
    tsmc_dla   (Q_buf, D, E, CDN, SDN, notifier);
    buf           (Q, Q_buf);
    not     (QN, Q_buf);

    specify
       (D => Q)=(0, 0);
       (D => QN)=(0, 0);
       (EN => Q)=(0, 0);
       (EN => QN)=(0, 0);
       $setup(posedge D, posedge EN , 0, notifier);
       $setup(negedge D, posedge EN , 0, notifier);
       $hold(posedge EN ,posedge  D, 0, notifier);
```

```
      $hold(posedge EN ,negedge  D, 0, notifier);
      $width(posedge EN, 0, 0, notifier);
      $width(negedge EN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LNx , Mon Nov 24 11:09:09 CST 1997
`timescale 1ns / 10ps
`celldefine
module LN3 (D, EN, Q, QN);

    input D, EN;
    output Q, QN;
    reg notifier;

    pullup (CDN);
    pullup (SDN);
    not    (E, EN);
    tsmc_dla   (Q_buf, D, E, CDN, SDN, notifier);
    buf          (Q, Q_buf);
    not    (QN, Q_buf);

    specify
      (D => Q)=(0, 0);
      (D => QN)=(0, 0);
      (EN => Q)=(0, 0);
      (EN => QN)=(0, 0);
      $setup(posedge D, posedge EN , 0, notifier);
      $setup(negedge D, posedge EN , 0, notifier);
```

```
        $hold(posedge EN ,posedge  D, 0, notifier);
        $hold(posedge EN ,negedge  D, 0, notifier);
        $width(posedge EN, 0, 0, notifier);
        $width(negedge EN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LNCNx , Mon Nov 24 11:09:09 CST 1997
`timescale 1ns / 10ps
`celldefine
module LNCN1 (D, EN, CDN, Q, QN);

    input D, EN, CDN;
    output Q, QN;
    reg notifier;

    buf           (CDN_i, CDN);
    pullup        (SDN);
    and           (EN_check, CDN_i, SDN);
    not     (E, EN);
    tsmc_dla   (Q_buf, D, E, CDN_i, SDN, notifier);
    buf           (Q, Q_buf);
    not     (QN, Q_buf);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (D => Q)=(0, 0);
       (D => QN)=(0, 0);
```

```
        (EN => Q)=(0, 0);
        (EN => QN)=(0, 0);
        $recovery(posedge CDN,  posedge EN, 0, notifier);
        $hold(posedge EN , posedge CDN, 0, notifier);
        $setup(posedge D, posedge EN &&& EN_check, 0, notifier);
        $setup(negedge D, posedge EN &&& EN_check, 0, notifier);
        $hold(posedge EN,posedge  D &&& EN_check, 0, notifier);
        $hold(posedge EN,negedge  D &&& EN_check, 0, notifier);
        $width(posedge EN &&& EN_check, 0, 0, notifier);
        $width(negedge EN &&& EN_check, 0, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LNCNx , Mon Nov 24 11:09:09 CST 1997
`timescale 1ns / 10ps
`celldefine
module LNCN2 (D, EN, CDN, Q, QN);

    input D, EN, CDN;
    output Q, QN;
    reg notifier;

    buf          (CDN_i, CDN);
    pullup       (SDN);
    and          (EN_check, CDN_i, SDN);
    not     (E, EN);
    tsmc_dla   (Q_buf, D, E, CDN_i, SDN, notifier);
```

```
    buf          (Q, Q_buf);
    not     (QN, Q_buf);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (D => Q)=(0, 0);
       (D => QN)=(0, 0);
       (EN => Q)=(0, 0);
       (EN => QN)=(0, 0);
       $recovery(posedge CDN,  posedge EN, 0, notifier);
       $hold(posedge EN , posedge CDN, 0, notifier);
       $setup(posedge D, posedge EN &&& EN_check, 0, notifier);
       $setup(negedge D, posedge EN &&& EN_check, 0, notifier);
       $hold(posedge EN,posedge  D &&& EN_check, 0, notifier);
       $hold(posedge EN,negedge  D &&& EN_check, 0, notifier);
       $width(posedge EN &&& EN_check, 0, 0, notifier);
       $width(negedge EN &&& EN_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LNCNx , Mon Nov 24 11:09:09 CST 1997
`timescale 1ns / 10ps
`celldefine
module LNCN3 (D, EN, CDN, Q, QN);

    input D, EN, CDN;
```

```
    output Q, QN;
    reg notifier;

    buf          (CDN_i, CDN);
    pullup       (SDN);
    and          (EN_check, CDN_i, SDN);
    not    (E, EN);
    tsmc_dla   (Q_buf, D, E, CDN_i, SDN, notifier);
    buf          (Q, Q_buf);
    not    (QN, Q_buf);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (D => Q)=(0, 0);
       (D => QN)=(0, 0);
       (EN => Q)=(0, 0);
       (EN => QN)=(0, 0);
       $recovery(posedge CDN,  posedge EN, 0, notifier);
       $hold(posedge EN , posedge CDN, 0, notifier);
       $setup(posedge D, posedge EN &&& EN_check, 0, notifier);
       $setup(negedge D, posedge EN &&& EN_check, 0, notifier);
       $hold(posedge EN,posedge  D &&& EN_check, 0, notifier);
       $hold(posedge EN,negedge  D &&& EN_check, 0, notifier);
       $width(posedge EN &&& EN_check, 0, 0, notifier);
       $width(negedge EN &&& EN_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: LNCSNx , Mon Nov 24 11:09:10 CST 1997
`timescale 1ns / 10ps
`celldefine
module LNCSN1 (D, EN, CDN, SDN, Q, QN);
   input D, EN, CDN, SDN;
   output Q, QN;
   reg notifier;

    buf          (CDN_i, CDN);
    buf          (SDN_i, SDN);
   and          (EN_check, CDN_i, SDN_i);
   not      (E, EN);
   tsmc_dla(Q_buf, D, E, CDN_i, SDN_i, notifier);
   buf          (Q, Q_buf);
   not      (QN_buf, Q_buf);
   and      (QN, QN_buf, SDN_i);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
         if (CDN_i!==1'b0) begin
         $display("%m > CDN is released at time %.2fns.", $realtime);
         end
         if (SDN_i!==1'b0) begin
         $display("%m > SDN is released at time %.2fns.", $realtime);
         end
      end
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
```

```
      if (flag == 1) begin
         $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end

   specify
      (CDN => Q)=(0, 0);
      (CDN => QN)=(0, 0);
      (D => Q)=(0, 0);
      (D => QN)=(0, 0);
      (EN => Q)=(0, 0);
      (EN => QN)=(0, 0);
      (SDN => Q)=(0, 0);
      (SDN => QN)=(0, 0);
       $recovery(posedge CDN,  posedge EN, 0, notifier);
       $hold(posedge EN , posedge CDN, 0, notifier);
       $recovery(posedge SDN,  posedge EN, 0, notifier);
       $hold(posedge EN , posedge SDN, 0, notifier);
       $setup(posedge D, posedge EN &&& EN_check , 0, notifier);
       $setup(negedge D, posedge EN &&& EN_check , 0, notifier);
       $hold(posedge EN,posedge  D &&& EN_check,  0, notifier);
       $hold(posedge EN,negedge  D &&& EN_check,  0, notifier);
       $width(posedge EN &&& EN_check, 0, 0, notifier);
       $width(negedge EN &&& EN_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
       $width(posedge SDN, 0, 0, notifier);
       $width(negedge SDN, 0, 0, notifier);
   endspecify
```

```
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LNCSNx , Mon Nov 24 11:09:10 CST 1997
`timescale 1ns / 10ps
`celldefine
module LNCSN2 (D, EN, CDN, SDN, Q, QN);
   input D, EN, CDN, SDN;
   output Q, QN;
   reg notifier;

    buf          (CDN_i, CDN);
    buf          (SDN_i, SDN);
   and          (EN_check, CDN_i, SDN_i);
   not      (E, EN);
   tsmc_dla(Q_buf, D, E, CDN_i, SDN_i, notifier);
   buf          (Q, Q_buf);
   not      (QN_buf, Q_buf);
   and      (QN, QN_buf, SDN_i);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
         if (CDN_i!==1'b0) begin
         $display("%m > CDN is released at time %.2fns.", $realtime);
         end
         if (SDN_i!==1'b0) begin
         $display("%m > SDN is released at time %.2fns.", $realtime);
         end
```

```
      end
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
      if (flag == 1) begin
         $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end

   specify
      (CDN => Q)=(0, 0);
      (CDN => QN)=(0, 0);
      (D => Q)=(0, 0);
      (D => QN)=(0, 0);
      (EN => Q)=(0, 0);
      (EN => QN)=(0, 0);
      (SDN => Q)=(0, 0);
      (SDN => QN)=(0, 0);
       $recovery(posedge CDN,  posedge EN, 0, notifier);
       $hold(posedge EN , posedge CDN, 0, notifier);
       $recovery(posedge SDN,  posedge EN, 0, notifier);
       $hold(posedge EN , posedge SDN, 0, notifier);
       $setup(posedge D, posedge EN &&& EN_check , 0, notifier);
       $setup(negedge D, posedge EN &&& EN_check , 0, notifier);
       $hold(posedge EN,posedge  D &&& EN_check,  0, notifier);
       $hold(posedge EN,negedge  D &&& EN_check,  0, notifier);
       $width(posedge EN &&& EN_check, 0, 0, notifier);
       $width(negedge EN &&& EN_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
       $width(posedge SDN, 0, 0, notifier);
```

```
       $width(negedge SDN, 0, 0, notifier);
   endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: LNCSNx , Mon Nov 24 11:09:10 CST 1997
`timescale 1ns / 10ps
`celldefine
module LNCSN3 (D, EN, CDN, SDN, Q, QN);
   input D, EN, CDN, SDN;
   output Q, QN;
   reg notifier;

    buf          (CDN_i, CDN);
    buf          (SDN_i, SDN);
   and          (EN_check, CDN_i, SDN_i);
   not      (E, EN);
   tsmc_dla(Q_buf, D, E, CDN_i, SDN_i, notifier);
   buf          (Q, Q_buf);
   not      (QN_buf, Q_buf);
   and      (QN, QN_buf, SDN_i);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
         if (CDN_i!==1'b0) begin
         $display("%m > CDN is released at time %.2fns.", $realtime);
         end
         if (SDN_i!==1'b0) begin
```

```
          $display("%m > SDN is released at time %.2fns.", $realtime);
          end
       end
       flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
       if (flag == 1) begin
          $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
       end
    end
    end

   specify
      (CDN => Q)=(0, 0);
      (CDN => QN)=(0, 0);
      (D => Q)=(0, 0);
      (D => QN)=(0, 0);
      (EN => Q)=(0, 0);
      (EN => QN)=(0, 0);
      (SDN => Q)=(0, 0);
      (SDN => QN)=(0, 0);
       $recovery(posedge CDN,  posedge EN, 0, notifier);
       $hold(posedge EN , posedge CDN, 0, notifier);
       $recovery(posedge SDN,  posedge EN, 0, notifier);
       $hold(posedge EN , posedge SDN, 0, notifier);
       $setup(posedge D, posedge EN &&& EN_check , 0, notifier);
       $setup(negedge D, posedge EN &&& EN_check , 0, notifier);
       $hold(posedge EN,posedge  D &&& EN_check,  0, notifier);
       $hold(posedge EN,negedge  D &&& EN_check,  0, notifier);
       $width(posedge EN &&& EN_check, 0, 0, notifier);
       $width(negedge EN &&& EN_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
```

```
        $width(negedge CDN, 0, 0, notifier);
        $width(posedge SDN, 0, 0, notifier);
        $width(negedge SDN, 0, 0, notifier);
   endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MAOI222Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module MAOI222D0 (A, B, C, ZN);

    input A, B, C;
    output ZN;

    and     (AB, A, B);
    and     (AC, A, C);
    and     (BC, B, C);
    nor     (ZN, AB, AC, BC);

    specify
       (A => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MAOI222Dx , Mon Nov 24 11:09:27 CST 1997
```

```
`timescale 1ns / 10ps
`celldefine
module MAOI222D1 (A, B, C, ZN);

    input A, B, C;
    output ZN;

    and    (AB, A, B);
    and    (AC, A, C);
    and    (BC, B, C);
    nor    (ZN, AB, AC, BC);

    specify
       (A => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MAOI222Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module MAOI222D2 (A, B, C, ZN);

    input A, B, C;
    output ZN;

    and    (AB, A, B);
    and    (AC, A, C);
```

```
    and     (BC, B, C);
    nor     (ZN, AB, AC, BC);

    specify
       (A => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MAOI22Dx , Mon Nov 24 11:09:28 CST 1997
`timescale 1ns / 10ps
`celldefine
module MAOI22D0 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    and     (A, A1, A2);
    nor     (B, B1, B2);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MAOI22Dx , Mon Nov 24 11:09:28 CST 1997
`timescale 1ns / 10ps
`celldefine
module MAOI22D1 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    and     (A, A1, A2);
    nor     (B, B1, B2);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MAOI22Dx , Mon Nov 24 11:09:28 CST 1997
`timescale 1ns / 10ps
`celldefine
module MAOI22D2 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
```

```
    output ZN;

    and     (A, A1, A2);
    nor     (B, B1, B2);
    nor     (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MOAI22Dx , Mon Nov 24 11:09:28 CST 1997
`timescale 1ns / 10ps
`celldefine
module MOAI22D0 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    or      (A, A1, A2);
    nand       (B, B1, B2);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
```

```
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MOAI22Dx , Mon Nov 24 11:09:28 CST 1997
`timescale 1ns / 10ps
`celldefine
module MOAI22D1 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    or      (A, A1, A2);
    nand        (B, B1, B2);
    nand        (ZN, A, B);

    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MOAI22Dx , Mon Nov 24 11:09:28 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module MOAI22D2 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    or      (A, A1, A2);
    nand       (B, B1, B2);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MUX2Dx , Wed Sep 23 08:57:35 CST 1998
`timescale 1ns / 10ps
`celldefine
module MUX2D1 (I0, I1, S, Z);

    input I0, I1, S;
    output Z;

    tsmc_mux (Z_buf, I0, I1, S);
    buf      (Z, Z_buf);
```

```
    specify
       (I0 => Z)=(0, 0);
       (I1 => Z)=(0, 0);
       (S => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MUX2Dx , Wed Sep 23 08:57:35 CST 1998
`timescale 1ns / 10ps
`celldefine
module MUX2D2 (I0, I1, S, Z);

    input I0, I1, S;
    output Z;

    tsmc_mux (Z_buf, I0, I1, S);
    buf      (Z, Z_buf);

    specify
       (I0 => Z)=(0, 0);
       (I1 => Z)=(0, 0);
       (S => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MUX4Dx , Wed Sep 23 08:57:35 CST 1998
`timescale 1ns / 10ps
```

```
`celldefine
module MUX4D1 (I0, I1, I2, I3, S0, S1, Z);

    input I0, I1, I2, I3, S0, S1;
    output Z;

    tsmc_mux (Z0, I0, I1, S0);
    tsmc_mux (Z1, I2, I3, S0);
    tsmc_mux (Z_buf, Z0, Z1, S1);
    buf      (Z, Z_buf);

    specify
        (I0 => Z)=(0, 0);
        (I1 => Z)=(0, 0);
        (I2 => Z)=(0, 0);
        (I3 => Z)=(0, 0);
        (S0 => Z)=(0, 0);
        (S1 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MUX4Dx , Wed Sep 23 08:57:35 CST 1998
`timescale 1ns / 10ps
`celldefine
module MUX4D2 (I0, I1, I2, I3, S0, S1, Z);

    input I0, I1, I2, I3, S0, S1;
    output Z;
```

```
    tsmc_mux (Z0, I0, I1, S0);
    tsmc_mux (Z1, I2, I3, S0);
    tsmc_mux (Z_buf, Z0, Z1, S1);
    buf      (Z, Z_buf);

    specify
        (I0 => Z)=(0, 0);
        (I1 => Z)=(0, 0);
        (I2 => Z)=(0, 0);
        (I3 => Z)=(0, 0);
        (S0 => Z)=(0, 0);
        (S1 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MUX8Dx , Wed Sep 23 08:57:35 CST 1998
`timescale 1ns / 10ps
`celldefine
module MUX8D1 (I0, I1, I2, I3, I4, I5, I6, I7, S0, S1, S2, Z);

    input I0, I1, I2, I3, I4, I5, I6, I7, S0, S1, S2;
    output Z;

    tsmc_mux (Z0, I0, I1, S0);
    tsmc_mux (Z1, I2, I3, S0);
    tsmc_mux (Z2, I4, I5, S0);
    tsmc_mux (Z3, I6, I7, S0);
    tsmc_mux (ZZ0, Z0, Z1, S1);
    tsmc_mux (ZZ1, Z2, Z3, S1);
```

```
    tsmc_mux (Z_buf, ZZ0, ZZ1, S2);
    buf      (Z, Z_buf);

    specify
       (I0 => Z)=(0, 0);
       (I1 => Z)=(0, 0);
       (I2 => Z)=(0, 0);
       (I3 => Z)=(0, 0);
       (I4 => Z)=(0, 0);
       (I5 => Z)=(0, 0);
       (I6 => Z)=(0, 0);
       (I7 => Z)=(0, 0);
       (S0 => Z)=(0, 0);
       (S1 => Z)=(0, 0);
       (S2 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MUX8Dx , Wed Sep 23 08:57:35 CST 1998
`timescale 1ns / 10ps
`celldefine
module MUX8D2 (I0, I1, I2, I3, I4, I5, I6, I7, S0, S1, S2, Z);

    input I0, I1, I2, I3, I4, I5, I6, I7, S0, S1, S2;
    output Z;

    tsmc_mux (Z0, I0, I1, S0);
    tsmc_mux (Z1, I2, I3, S0);
    tsmc_mux (Z2, I4, I5, S0);
```

```
    tsmc_mux (Z3, I6, I7, S0);
    tsmc_mux (ZZ0, Z0, Z1, S1);
    tsmc_mux (ZZ1, Z2, Z3, S1);
    tsmc_mux (Z_buf, ZZ0, ZZ1, S2);
    buf      (Z, Z_buf);

    specify
       (I0 => Z)=(0, 0);
       (I1 => Z)=(0, 0);
       (I2 => Z)=(0, 0);
       (I3 => Z)=(0, 0);
       (I4 => Z)=(0, 0);
       (I5 => Z)=(0, 0);
       (I6 => Z)=(0, 0);
       (I7 => Z)=(0, 0);
       (S0 => Z)=(0, 0);
       (S1 => Z)=(0, 0);
       (S2 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: MUXN4Dx , Wed Sep 23 08:57:35 CST 1998
`timescale 1ns / 10ps
`celldefine
module MUXN4D1 (I0, I1, I2, I3, S0, S1, EN, Z);

    input I0, I1, I2, I3, S0, S1, EN;
    output Z;
```

```
    not       (ENB, EN);
    tsmc_mux (Z0, I0, I1, S0);
    tsmc_mux (Z1, I2, I3, S0);
    tsmc_mux (Z_buf, Z0, Z1, S1);
    and       (Z, Z_buf, ENB);

    specify
       (I0 => Z)=(0, 0);
       (I1 => Z)=(0, 0);
       (I2 => Z)=(0, 0);
       (I3 => Z)=(0, 0);
       (S0 => Z)=(0, 0);
       (S1 => Z)=(0, 0);
       (EN => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND2Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND2D0 (A1, A2, ZN);
    input A1, A2;
    output ZN;
    nand       (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND2Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND2D1 (A1, A2, ZN);
    input A1, A2;
    output ZN;
    nand       (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND2Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND2D1H (A1, A2, ZN);
    input A1, A2;
    output ZN;
    nand       (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: ND2Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND2D2 (A1, A2, ZN);
    input A1, A2;
    output ZN;
    nand       (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND2Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND2D3 (A1, A2, ZN);
    input A1, A2;
    output ZN;
    nand       (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: ND3Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND3D0 (A1, A2, A3, ZN);
    input A1, A2, A3;
    output ZN;
    nand       (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND3Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND3D1 (A1, A2, A3, ZN);
    input A1, A2, A3;
    output ZN;
    nand       (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: ND3Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND3D1H (A1, A2, A3, ZN);
    input A1, A2, A3;
    output ZN;
    nand      (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND3Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND3D2 (A1, A2, A3, ZN);
    input A1, A2, A3;
    output ZN;
    nand      (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND3Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND3D3 (A1, A2, A3, ZN);
    input A1, A2, A3;
    output ZN;
    nand       (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND4Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND4D0 (A1, A2, A3, A4, ZN);
    input A1, A2, A3, A4;
    output ZN;
    nand       (ZN, A1, A2, A3, A4);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
```

```
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND4Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND4D1 (A1, A2, A3, A4, ZN);
    input A1, A2, A3, A4;
    output ZN;
    nand       (ZN, A1, A2, A3, A4);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND4Dx , Mon Nov 24 11:08:47 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND4D1H (A1, A2, A3, A4, ZN);
    input A1, A2, A3, A4;
    output ZN;
    nand       (ZN, A1, A2, A3, A4);
    specify
       (A1 => ZN)=(0, 0);
```

```
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (A4 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND4Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND4D2 (A1, A2, A3, A4, ZN);
    input A1, A2, A3, A4;
    output ZN;
    nand       (ZN, A1, A2, A3, A4);
    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (A4 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND4Dx , Mon Nov 24 11:08:46 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND4D3 (A1, A2, A3, A4, ZN);
    input A1, A2, A3, A4;
    output ZN;
```

```
    nand       (ZN, A1, A2, A3, A4);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND5Dx , Mon Nov 24 11:08:47 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND5D1 (A1, A2, A3, A4, A5, ZN);
    input A1, A2, A3, A4, A5;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND5Dx , Mon Nov 24 11:08:47 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module ND5D2 (A1, A2, A3, A4, A5, ZN);
    input A1, A2, A3, A4, A5;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND5Dx , Mon Nov 24 11:08:47 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND5D3 (A1, A2, A3, A4, A5, ZN);
    input A1, A2, A3, A4, A5;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND6Dx , Mon Nov 24 11:08:47 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND6D1 (A1, A2, A3, A4, A5, A6, ZN);
    input A1, A2, A3, A4, A5, A6;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5, A6);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
       (A6 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND6Dx , Mon Nov 24 11:08:47 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND6D2 (A1, A2, A3, A4, A5, A6, ZN);
    input A1, A2, A3, A4, A5, A6;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5, A6);
    specify
       (A1 => ZN)=(0, 0);
```

```
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (A4 => ZN)=(0, 0);
        (A5 => ZN)=(0, 0);
        (A6 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND6Dx , Mon Nov 24 11:08:47 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND6D3 (A1, A2, A3, A4, A5, A6, ZN);
    input A1, A2, A3, A4, A5, A6;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5, A6);
    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (A4 => ZN)=(0, 0);
        (A5 => ZN)=(0, 0);
        (A6 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND7Dx , Mon Nov 24 11:08:47 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module ND7D1 (A1, A2, A3, A4, A5, A6, A7, ZN);
    input A1, A2, A3, A4, A5, A6, A7;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5, A6, A7);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
       (A6 => ZN)=(0, 0);
       (A7 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND7Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND7D2 (A1, A2, A3, A4, A5, A6, A7, ZN);
    input A1, A2, A3, A4, A5, A6, A7;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5, A6, A7);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
```

```
        (A6 => ZN)=(0, 0);
        (A7 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND7Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND7D3 (A1, A2, A3, A4, A5, A6, A7, ZN);
    input A1, A2, A3, A4, A5, A6, A7;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5, A6, A7);
    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (A4 => ZN)=(0, 0);
        (A5 => ZN)=(0, 0);
        (A6 => ZN)=(0, 0);
        (A7 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND8Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND8D1 (A1, A2, A3, A4, A5, A6, A7, A8, ZN);
```

```
    input A1, A2, A3, A4, A5, A6, A7, A8;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5, A6, A7, A8);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
       (A6 => ZN)=(0, 0);
       (A7 => ZN)=(0, 0);
       (A8 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND8Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND8D2 (A1, A2, A3, A4, A5, A6, A7, A8, ZN);
    input A1, A2, A3, A4, A5, A6, A7, A8;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5, A6, A7, A8);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
       (A6 => ZN)=(0, 0);
```

```
        (A7 => ZN)=(0, 0);
        (A8 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: ND8Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
`celldefine
module ND8D3 (A1, A2, A3, A4, A5, A6, A7, A8, ZN);
    input A1, A2, A3, A4, A5, A6, A7, A8;
    output ZN;
    nand       (ZN, A1, A2, A3, A4, A5, A6, A7, A8);
    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (A4 => ZN)=(0, 0);
        (A5 => ZN)=(0, 0);
        (A6 => ZN)=(0, 0);
        (A7 => ZN)=(0, 0);
        (A8 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NDLx , Mon Nov 24 11:09:05 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module NDL1 (RN, SN, Q, QN);

    input RN, SN;
    output Q, QN;
    reg notifier;

    tsmc_ndla  (Q_buf, RN, SN, notifier);
    tsmc_ndla  (QN_buf, SN, RN, notifier);
    not    (Q, QN_buf);
    not    (QN, Q_buf);

    specify
       (RN => Q)=(0, 0);
       (RN => QN)=(0, 0);
       (SN => Q)=(0, 0);
       (SN => QN)=(0, 0);
       $setup(posedge RN, posedge SN , 0, notifier);
       $hold(posedge SN , posedge RN, 0, notifier);
       $setup(posedge SN, posedge RN , 0, notifier);
       $hold(posedge RN , posedge SN, 0, notifier);
       $width(posedge RN, 0, 0, notifier);
       $width(negedge RN, 0, 0, notifier);
       $width(posedge SN, 0, 0, notifier);
       $width(negedge SN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NDLx , Mon Nov 24 11:09:05 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module NDL2 (RN, SN, Q, QN);

    input RN, SN;
    output Q, QN;
    reg notifier;

    tsmc_ndla  (Q_buf, RN, SN, notifier);
    tsmc_ndla  (QN_buf, SN, RN, notifier);
    not    (Q, QN_buf);
    not    (QN, Q_buf);

    specify
       (RN => Q)=(0, 0);
       (RN => QN)=(0, 0);
       (SN => Q)=(0, 0);
       (SN => QN)=(0, 0);
       $setup(posedge RN, posedge SN , 0, notifier);
       $hold(posedge SN , posedge RN, 0, notifier);
       $setup(posedge SN, posedge RN , 0, notifier);
       $hold(posedge RN , posedge SN, 0, notifier);
       $width(posedge RN, 0, 0, notifier);
       $width(negedge RN, 0, 0, notifier);
       $width(posedge SN, 0, 0, notifier);
       $width(negedge SN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR2Dx , Mon Nov 24 11:08:48 CST 1997
```

```
`timescale 1ns / 10ps
`celldefine
module NR2D0 (A1, A2, ZN);
    input A1, A2;
    output ZN;
    nor    (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR2Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR2D1 (A1, A2, ZN);
    input A1, A2;
    output ZN;
    nor    (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR2Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module NR2D1H (A1, A2, ZN);
    input A1, A2;
    output ZN;
    nor    (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR2Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR2D2 (A1, A2, ZN);
    input A1, A2;
    output ZN;
    nor    (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR2Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module NR2D3 (A1, A2, ZN);
    input A1, A2;
    output ZN;
    nor    (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR3Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR3D0 (A1, A2, A3, ZN);
    input A1, A2, A3;
    output ZN;
    nor    (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR3Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module NR3D1 (A1, A2, A3, ZN);
    input A1, A2, A3;
    output ZN;
    nor    (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR3Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR3D1H (A1, A2, A3, ZN);
    input A1, A2, A3;
    output ZN;
    nor    (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR3Dx , Mon Nov 24 11:08:48 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module NR3D2 (A1, A2, A3, ZN);
    input A1, A2, A3;
    output ZN;
    nor    (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR3Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR3D3 (A1, A2, A3, ZN);
    input A1, A2, A3;
    output ZN;
    nor    (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR4Dx , Mon Nov 24 11:08:49 CST 1997
```

```
`timescale 1ns / 10ps
`celldefine
module NR4D0 (A1, A2, A3, A4, ZN);
    input A1, A2, A3, A4;
    output ZN;
    nor    (ZN, A1, A2, A3, A4);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR4Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR4D1 (A1, A2, A3, A4, ZN);
    input A1, A2, A3, A4;
    output ZN;
    nor    (ZN, A1, A2, A3, A4);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: NR4Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR4D1H (A1, A2, A3, A4, ZN);
    input A1, A2, A3, A4;
    output ZN;
    nor    (ZN, A1, A2, A3, A4);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR4Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR4D2 (A1, A2, A3, A4, ZN);
    input A1, A2, A3, A4;
    output ZN;
    nor    (ZN, A1, A2, A3, A4);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
```

```
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR4Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR4D3 (A1, A2, A3, A4, ZN);
    input A1, A2, A3, A4;
    output ZN;
    nor    (ZN, A1, A2, A3, A4);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR5Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR5D1 (A1, A2, A3, A4, A5, ZN);
    input A1, A2, A3, A4, A5;
    output ZN;
    nor    (ZN, A1, A2, A3, A4, A5);
    specify
       (A1 => ZN)=(0, 0);
```

```
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (A4 => ZN)=(0, 0);
        (A5 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR5Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR5D2 (A1, A2, A3, A4, A5, ZN);
    input A1, A2, A3, A4, A5;
    output ZN;
    nor    (ZN, A1, A2, A3, A4, A5);
    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (A4 => ZN)=(0, 0);
        (A5 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR5Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR5D3 (A1, A2, A3, A4, A5, ZN);
```

```
    input A1, A2, A3, A4, A5;
    output ZN;
    nor    (ZN, A1, A2, A3, A4, A5);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR6Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR6D1 (A1, A2, A3, A4, A5, A6, ZN);
    input A1, A2, A3, A4, A5, A6;
    output ZN;
    nor    (ZN, A1, A2, A3, A4, A5, A6);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
       (A6 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: NR6Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR6D2 (A1, A2, A3, A4, A5, A6, ZN);
    input A1, A2, A3, A4, A5, A6;
    output ZN;
    nor    (ZN, A1, A2, A3, A4, A5, A6);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
       (A6 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR6Dx , Mon Nov 24 11:08:49 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR6D3 (A1, A2, A3, A4, A5, A6, ZN);
    input A1, A2, A3, A4, A5, A6;
    output ZN;
    nor    (ZN, A1, A2, A3, A4, A5, A6);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
```

```
        (A3 => ZN)=(0, 0);
        (A4 => ZN)=(0, 0);
        (A5 => ZN)=(0, 0);
        (A6 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR7Dx , Mon Nov 24 11:08:50 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR7D1 (A1, A2, A3, A4, A5, A6, A7, ZN);
    input A1, A2, A3, A4, A5, A6, A7;
    output ZN;
    nor    (ZN, A1, A2, A3, A4, A5, A6, A7);
    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (A4 => ZN)=(0, 0);
        (A5 => ZN)=(0, 0);
        (A6 => ZN)=(0, 0);
        (A7 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR7Dx , Mon Nov 24 11:08:50 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module NR7D2 (A1, A2, A3, A4, A5, A6, A7, ZN);
    input A1, A2, A3, A4, A5, A6, A7;
    output ZN;
    nor    (ZN, A1, A2, A3, A4, A5, A6, A7);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
       (A6 => ZN)=(0, 0);
       (A7 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR7Dx , Mon Nov 24 11:08:50 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR7D3 (A1, A2, A3, A4, A5, A6, A7, ZN);
    input A1, A2, A3, A4, A5, A6, A7;
    output ZN;
    nor    (ZN, A1, A2, A3, A4, A5, A6, A7);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
```

```
        (A6 => ZN)=(0, 0);
        (A7 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR8Dx , Mon Nov 24 11:08:50 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR8D1 (A1, A2, A3, A4, A5, A6, A7, A8, ZN);
    input A1, A2, A3, A4, A5, A6, A7, A8;
    output ZN;
    nor   (ZN, A1, A2, A3, A4, A5, A6, A7, A8);
    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (A4 => ZN)=(0, 0);
        (A5 => ZN)=(0, 0);
        (A6 => ZN)=(0, 0);
        (A7 => ZN)=(0, 0);
        (A8 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR8Dx , Mon Nov 24 11:08:50 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module NR8D2 (A1, A2, A3, A4, A5, A6, A7, A8, ZN);
    input A1, A2, A3, A4, A5, A6, A7, A8;
    output ZN;
    nor    (ZN, A1, A2, A3, A4, A5, A6, A7, A8);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
       (A6 => ZN)=(0, 0);
       (A7 => ZN)=(0, 0);
       (A8 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NR8Dx , Mon Nov 24 11:08:50 CST 1997
`timescale 1ns / 10ps
`celldefine
module NR8D3 (A1, A2, A3, A4, A5, A6, A7, A8, ZN);
    input A1, A2, A3, A4, A5, A6, A7, A8;
    output ZN;
    nor    (ZN, A1, A2, A3, A4, A5, A6, A7, A8);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (A4 => ZN)=(0, 0);
       (A5 => ZN)=(0, 0);
```

```
        (A6 => ZN)=(0, 0);
        (A7 => ZN)=(0, 0);
        (A8 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NRLx , Mon Nov 24 11:09:06 CST 1997
`timescale 1ns / 10ps
`celldefine
module NRL1 (R, S, Q, QN);

    input R, S;
    output Q, QN;
    reg notifier;

    tsmc_nrla  (Q_buf, R, S, notifier);
    tsmc_nrla  (QN_buf, S, R, notifier);
    not    (Q, QN_buf);
    not    (QN, Q_buf);

    specify
       (R => Q)=(0, 0);
       (R => QN)=(0, 0);
       (S => Q)=(0, 0);
       (S => QN)=(0, 0);
       $setup(negedge R, negedge S , 0, notifier);
       $hold(negedge S , negedge R, 0, notifier);
       $setup(negedge S, negedge R , 0, notifier);
       $hold(negedge R , negedge S, 0, notifier);
```

```
        $width(posedge R, 0, 0, notifier);
        $width(negedge R, 0, 0, notifier);
        $width(posedge S, 0, 0, notifier);
        $width(negedge S, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: NRLx , Mon Nov 24 11:09:06 CST 1997
`timescale 1ns / 10ps
`celldefine
module NRL2 (R, S, Q, QN);

    input R, S;
    output Q, QN;
    reg notifier;

    tsmc_nrla  (Q_buf, R, S, notifier);
    tsmc_nrla  (QN_buf, S, R, notifier);
    not    (Q, QN_buf);
    not    (QN, Q_buf);

    specify
        (R => Q)=(0, 0);
        (R => QN)=(0, 0);
        (S => Q)=(0, 0);
        (S => QN)=(0, 0);
        $setup(negedge R, negedge S , 0, notifier);
        $hold(negedge S , negedge R, 0, notifier);
        $setup(negedge S, negedge R , 0, notifier);
```

```
       $hold(negedge R , negedge S, 0, notifier);
       $width(posedge R, 0, 0, notifier);
       $width(negedge R, 0, 0, notifier);
       $width(posedge S, 0, 0, notifier);
       $width(negedge S, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI211Dx , Mon Nov 24 11:09:25 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI211D0 (A1, A2, B, C, ZN);

    input A1, A2, B, C;
    output ZN;

    or      (A, A1, A2);
    nand       (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
```

```
// Model Type: OAI211Dx , Mon Nov 24 11:09:25 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI211D1 (A1, A2, B, C, ZN);

    input A1, A2, B, C;
    output ZN;

    or     (A, A1, A2);
    nand       (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI211Dx , Mon Nov 24 11:09:25 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI211D1H (A1, A2, B, C, ZN);

    input A1, A2, B, C;
    output ZN;

    or     (A, A1, A2);
    nand       (ZN, A, B, C);
```

```
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI211Dx , Mon Nov 24 11:09:25 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI211D2 (A1, A2, B, C, ZN);

    input A1, A2, B, C;
    output ZN;

    or      (A, A1, A2);
    nand        (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: OAI211Dx , Mon Nov 24 11:09:25 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI211D3 (A1, A2, B, C, ZN);

    input A1, A2, B, C;
    output ZN;

    or      (A, A1, A2);
    nand        (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI21Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI21D0 (A1, A2, B, ZN);
    input A1, A2, B;
    output ZN;
    or      (A, A1, A2);
    nand        (ZN, A, B);
```

```
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI21Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI21D1 (A1, A2, B, ZN);
    input A1, A2, B;
    output ZN;
    or     (A, A1, A2);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI21Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI21D1H (A1, A2, B, ZN);
```

```
    input A1, A2, B;
    output ZN;
    or      (A, A1, A2);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI21Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI21D2 (A1, A2, B, ZN);
    input A1, A2, B;
    output ZN;
    or      (A, A1, A2);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: OAI21Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI21D3 (A1, A2, B, ZN);
    input A1, A2, B;
    output ZN;
    or      (A, A1, A2);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI221Dx , Mon Nov 24 11:09:25 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI221D1 (A1, A2, B1, B2, C, ZN);

    input A1, A2, B1, B2, C;
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
    nand         (ZN, A, B, C);
    specify
```

```
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
        (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI221Dx , Mon Nov 24 11:09:25 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI221D1H (A1, A2, B1, B2, C, ZN);

    input A1, A2, B1, B2, C;
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
    nand          (ZN, A, B, C);
    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
        (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: OAI221Dx , Mon Nov 24 11:09:25 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI221D2 (A1, A2, B1, B2, C, ZN);

    input A1, A2, B1, B2, C;
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
    nand          (ZN, A, B, C);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI221Dx , Mon Nov 24 11:09:25 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI221D3 (A1, A2, B1, B2, C, ZN);

    input A1, A2, B1, B2, C;
    output ZN;
```

```
    or      (A, A1, A2);
    or      (B, B1, B2);
    nand          (ZN, A, B, C);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (C => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI222Dx , Mon Nov 24 11:09:25 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI222D1 (A1, A2, B1, B2, C1, C2, ZN);

    input A1, A2, B1, B2, C1, C2;
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
    or      (C, C1, C2);
    nand       (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
```

```
        (B2 => ZN)=(0, 0);
        (C1 => ZN)=(0, 0);
        (C2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI222Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI222D1H (A1, A2, B1, B2, C1, C2, ZN);

    input A1, A2, B1, B2, C1, C2;
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
    or      (C, C1, C2);
    nand        (ZN, A, B, C);

    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
        (C1 => ZN)=(0, 0);
        (C2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: OAI222Dx , Mon Nov 24 11:09:25 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI222D2 (A1, A2, B1, B2, C1, C2, ZN);

    input A1, A2, B1, B2, C1, C2;
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
    or      (C, C1, C2);
    nand        (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (C1 => ZN)=(0, 0);
       (C2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI222Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI222D3 (A1, A2, B1, B2, C1, C2, ZN);
```

```
    input A1, A2, B1, B2, C1, C2;
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
    or      (C, C1, C2);
    nand        (ZN, A, B, C);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (C1 => ZN)=(0, 0);
       (C2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI22Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI22D0 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
```

```
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI22Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI22D1 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI22Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI22D1H (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI22Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI22D2 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
```

```
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
    nand        (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI22Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI22D3 (A1, A2, B1, B2, ZN);

    input A1, A2, B1, B2;
    output ZN;

    or      (A, A1, A2);
    or      (B, B1, B2);
    nand        (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
```

```
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI31Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI31D0 (A1, A2, A3, B, ZN);

    input A1, A2, A3, B;
    output ZN;

    or      (A, A1, A2, A3);
    nand        (ZN, A, B);

    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI31Dx , Mon Nov 24 11:09:26 CST 1997
`timescale 1ns / 10ps
`celldefine
```

```
module OAI31D1 (A1, A2, A3, B, ZN);

    input A1, A2, A3, B;
    output ZN;

    or      (A, A1, A2, A3);
    nand        (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI31Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI31D1H (A1, A2, A3, B, ZN);

    input A1, A2, A3, B;
    output ZN;

    or      (A, A1, A2, A3);
    nand        (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
```

```
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI31Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI31D2 (A1, A2, A3, B, ZN);

    input A1, A2, A3, B;
    output ZN;

    or      (A, A1, A2, A3);
    nand       (ZN, A, B);

    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI31Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
```

```
`celldefine
module OAI31D3 (A1, A2, A3, B, ZN);

    input A1, A2, A3, B;
    output ZN;

    or      (A, A1, A2, A3);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI32Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI32D0 (A1, A2, A3, B1, B2, ZN);

    input A1, A2, A3, B1, B2;
    output ZN;

    or      (A, A1, A2, A3);
    or      (B, B1, B2);
    nand       (ZN, A, B);
```

```
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI32Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI32D1 (A1, A2, A3, B1, B2, ZN);

    input A1, A2, A3, B1, B2;
    output ZN;

    or      (A, A1, A2, A3);
    or      (B, B1, B2);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI32Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI32D1H (A1, A2, A3, B1, B2, ZN);

    input A1, A2, A3, B1, B2;
    output ZN;

    or      (A, A1, A2, A3);
    or      (B, B1, B2);
    nand        (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI32Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI32D2 (A1, A2, A3, B1, B2, ZN);
```

```
    input A1, A2, A3, B1, B2;
    output ZN;

    or      (A, A1, A2, A3);
    or      (B, B1, B2);
    nand        (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI32Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI32D3 (A1, A2, A3, B1, B2, ZN);

    input A1, A2, A3, B1, B2;
    output ZN;

    or      (A, A1, A2, A3);
    or      (B, B1, B2);
    nand        (ZN, A, B);

    specify
```

```
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI33Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI33D0 (A1, A2, A3, B1, B2, B3, ZN);

    input A1, A2, A3, B1, B2, B3;
    output ZN;

    or      (A, A1, A2, A3);
    or      (B, B1, B2, B3);
    nand        (ZN, A, B);

    specify
        (A1 => ZN)=(0, 0);
        (A2 => ZN)=(0, 0);
        (A3 => ZN)=(0, 0);
        (B1 => ZN)=(0, 0);
        (B2 => ZN)=(0, 0);
        (B3 => ZN)=(0, 0);
    endspecify
endmodule
```

```
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI33Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI33D1 (A1, A2, A3, B1, B2, B3, ZN);

    input A1, A2, A3, B1, B2, B3;
    output ZN;

    or      (A, A1, A2, A3);
    or      (B, B1, B2, B3);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (B3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI33Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI33D1H (A1, A2, A3, B1, B2, B3, ZN);
```

```
    input A1, A2, A3, B1, B2, B3;
    output ZN;

    or      (A, A1, A2, A3);
    or      (B, B1, B2, B3);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (B3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI33Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI33D2 (A1, A2, A3, B1, B2, B3, ZN);

    input A1, A2, A3, B1, B2, B3;
    output ZN;

    or      (A, A1, A2, A3);
    or      (B, B1, B2, B3);
    nand       (ZN, A, B);
```

```
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
       (B3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OAI33Dx , Mon Nov 24 11:09:27 CST 1997
`timescale 1ns / 10ps
`celldefine
module OAI33D3 (A1, A2, A3, B1, B2, B3, ZN);

    input A1, A2, A3, B1, B2, B3;
    output ZN;

    or      (A, A1, A2, A3);
    or      (B, B1, B2, B3);
    nand       (ZN, A, B);

    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
       (B1 => ZN)=(0, 0);
       (B2 => ZN)=(0, 0);
```

```
       (B3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR2Dx , Mon Nov 24 11:08:50 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR2D1 (A1, A2, Z);
    input A1, A2;
    output Z;
    or      (Z, A1, A2);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR2Dx , Mon Nov 24 11:08:50 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR2D2 (A1, A2, Z);
    input A1, A2;
    output Z;
    or      (Z, A1, A2);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
```

```
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR2Dx , Mon Nov 24 11:08:51 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR2D3 (A1, A2, Z);
    input A1, A2;
    output Z;
    or      (Z, A1, A2);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR3Dx , Mon Nov 24 11:08:51 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR3D1 (A1, A2, A3, Z);
    input A1, A2, A3;
    output Z;
    or      (Z, A1, A2, A3);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
```

```
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR3Dx , Mon Nov 24 11:08:51 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR3D2 (A1, A2, A3, Z);
    input A1, A2, A3;
    output Z;
    or     (Z, A1, A2, A3);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR3Dx , Mon Nov 24 11:08:51 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR3D3 (A1, A2, A3, Z);
    input A1, A2, A3;
    output Z;
    or     (Z, A1, A2, A3);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
```

```
       (A3 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR4Dx , Mon Nov 24 11:08:51 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR4D1 (A1, A2, A3, A4, Z);
    input A1, A2, A3, A4;
    output Z;
    or      (Z, A1, A2, A3, A4);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR4Dx , Mon Nov 24 11:08:51 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR4D2 (A1, A2, A3, A4, Z);
    input A1, A2, A3, A4;
    output Z;
    or      (Z, A1, A2, A3, A4);
    specify
```

```
        (A1 => Z)=(0, 0);
        (A2 => Z)=(0, 0);
        (A3 => Z)=(0, 0);
        (A4 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR4Dx , Mon Nov 24 11:08:51 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR4D3 (A1, A2, A3, A4, Z);
    input A1, A2, A3, A4;
    output Z;
    or      (Z, A1, A2, A3, A4);
    specify
        (A1 => Z)=(0, 0);
        (A2 => Z)=(0, 0);
        (A3 => Z)=(0, 0);
        (A4 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR5Dx , Mon Nov 24 11:08:52 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR5D1 (A1, A2, A3, A4, A5, Z);
    input A1, A2, A3, A4, A5;
```

```
    output Z;
    or      (Z, A1, A2, A3, A4, A5);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR5Dx , Mon Nov 24 11:08:52 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR5D2 (A1, A2, A3, A4, A5, Z);
    input A1, A2, A3, A4, A5;
    output Z;
    or      (Z, A1, A2, A3, A4, A5);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
```

```
// Model Type: OR5Dx , Mon Nov 24 11:08:52 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR5D3 (A1, A2, A3, A4, A5, Z);
    input A1, A2, A3, A4, A5;
    output Z;
    or      (Z, A1, A2, A3, A4, A5);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR6Dx , Mon Nov 24 11:08:52 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR6D1 (A1, A2, A3, A4, A5, A6, Z);
    input A1, A2, A3, A4, A5, A6;
    output Z;
    or      (Z, A1, A2, A3, A4, A5, A6);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
```

```
        (A6 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR6Dx , Mon Nov 24 11:08:52 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR6D2 (A1, A2, A3, A4, A5, A6, Z);
    input A1, A2, A3, A4, A5, A6;
    output Z;
    or     (Z, A1, A2, A3, A4, A5, A6);
    specify
        (A1 => Z)=(0, 0);
        (A2 => Z)=(0, 0);
        (A3 => Z)=(0, 0);
        (A4 => Z)=(0, 0);
        (A5 => Z)=(0, 0);
        (A6 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR6Dx , Mon Nov 24 11:08:52 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR6D3 (A1, A2, A3, A4, A5, A6, Z);
    input A1, A2, A3, A4, A5, A6;
    output Z;
```

```
    or      (Z, A1, A2, A3, A4, A5, A6);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
       (A6 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR7Dx , Mon Nov 24 11:08:52 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR7D1 (A1, A2, A3, A4, A5, A6, A7, Z);
    input A1, A2, A3, A4, A5, A6, A7;
    output Z;
    or      (Z, A1, A2, A3, A4, A5, A6, A7);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
       (A6 => Z)=(0, 0);
       (A7 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: OR7Dx , Mon Nov 24 11:08:52 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR7D2 (A1, A2, A3, A4, A5, A6, A7, Z);
    input A1, A2, A3, A4, A5, A6, A7;
    output Z;
    or     (Z, A1, A2, A3, A4, A5, A6, A7);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
       (A6 => Z)=(0, 0);
       (A7 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR7Dx , Mon Nov 24 11:08:52 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR7D3 (A1, A2, A3, A4, A5, A6, A7, Z);
    input A1, A2, A3, A4, A5, A6, A7;
    output Z;
    or     (Z, A1, A2, A3, A4, A5, A6, A7);
    specify
       (A1 => Z)=(0, 0);
```

```
        (A2 => Z)=(0, 0);
        (A3 => Z)=(0, 0);
        (A4 => Z)=(0, 0);
        (A5 => Z)=(0, 0);
        (A6 => Z)=(0, 0);
        (A7 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR8Dx , Mon Nov 24 11:08:53 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR8D1 (A1, A2, A3, A4, A5, A6, A7, A8, Z);
    input A1, A2, A3, A4, A5, A6, A7, A8;
    output Z;
    or     (Z, A1, A2, A3, A4, A5, A6, A7, A8);
    specify
        (A1 => Z)=(0, 0);
        (A2 => Z)=(0, 0);
        (A3 => Z)=(0, 0);
        (A4 => Z)=(0, 0);
        (A5 => Z)=(0, 0);
        (A6 => Z)=(0, 0);
        (A7 => Z)=(0, 0);
        (A8 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine
```

```
// TSMC Standard Cell Function Library Ver
// Model Type: OR8Dx , Mon Nov 24 11:08:53 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR8D2 (A1, A2, A3, A4, A5, A6, A7, A8, Z);
    input A1, A2, A3, A4, A5, A6, A7, A8;
    output Z;
    or      (Z, A1, A2, A3, A4, A5, A6, A7, A8);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
       (A4 => Z)=(0, 0);
       (A5 => Z)=(0, 0);
       (A6 => Z)=(0, 0);
       (A7 => Z)=(0, 0);
       (A8 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: OR8Dx , Mon Nov 24 11:08:53 CST 1997
`timescale 1ns / 10ps
`celldefine
module OR8D3 (A1, A2, A3, A4, A5, A6, A7, A8, Z);
    input A1, A2, A3, A4, A5, A6, A7, A8;
    output Z;
    or      (Z, A1, A2, A3, A4, A5, A6, A7, A8);
    specify
       (A1 => Z)=(0, 0);
```

```
        (A2 => Z)=(0, 0);
        (A3 => Z)=(0, 0);
        (A4 => Z)=(0, 0);
        (A5 => Z)=(0, 0);
        (A6 => Z)=(0, 0);
        (A7 => Z)=(0, 0);
        (A8 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: TFCNx , Mon Nov 24 11:09:05 CST 1997
`timescale 1ns / 10ps
`celldefine
module TFCN1 (CP, CDN, Q, QN);

    input CP, CDN;
    output Q, QN;
    reg notifier;

    buf          (CDN_i, CDN);
    pullup (SDN);
    and     (CP_check, CDN_i, SDN);
    not     (D, Q_buf);
    tsmc_dff   (Q_buf, D, CP, CDN_i, SDN, notifier);
    buf     (Q, Q_buf);
    not     (QN_buf, Q_buf);
    and     (QN, QN_buf, SDN);

    specify
```

```
        (CDN => Q)=(0, 0);
        (CDN => QN)=(0, 0);
        (CP => Q)=(0, 0);
        (CP => QN)=(0, 0);
        $recovery(posedge CDN,  posedge CP, 0, notifier);
        $hold(posedge CP , posedge CDN, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: TFCNx , Mon Nov 24 11:09:05 CST 1997
`timescale 1ns / 10ps
`celldefine
module TFCN2 (CP, CDN, Q, QN);

    input CP, CDN;
    output Q, QN;
    reg notifier;

    buf          (CDN_i, CDN);
    pullup (SDN);
    and    (CP_check, CDN_i, SDN);
    not    (D, Q_buf);
    tsmc_dff   (Q_buf, D, CP, CDN_i, SDN, notifier);
    buf    (Q, Q_buf);
    not    (QN_buf, Q_buf);
```

```
    and     (QN, QN_buf, SDN);

    specify
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: TFCSNx , Mon Nov 24 11:09:05 CST 1997
`timescale 1ns / 10ps
`celldefine
module TFCSN1 (CP, CDN, SDN, Q, QN);

    input CP, CDN, SDN;
    output Q, QN;
    reg notifier;

    buf           (CDN_i, CDN);
    buf           (SDN_i, SDN);
    and     (CP_check, CDN_i, SDN_i);
    not     (D, Q_buf);
```

```
    tsmc_dff   (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf         (Q, Q_buf);
    not     (QN_buf, Q_buf);
    and     (QN, QN_buf, SDN_i);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
         if (CDN_i!==1'b0) begin
         $display("%m > CDN is released at time %.2fns.", $realtime);
         end
         if (SDN_i!==1'b0) begin
         $display("%m > SDN is released at time %.2fns.", $realtime);
         end
      end
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
      if (flag == 1) begin
         $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (SDN => Q)=(0, 0);
       (SDN => QN)=(0, 0);
```

```
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge SDN, 0, notifier);
       $width(posedge CP &&& CP_check, 0, 0, notifier);
       $width(negedge CP &&& CP_check, 0, 0, notifier);
       $width(posedge CDN, 0, 0, notifier);
       $width(negedge CDN, 0, 0, notifier);
       $width(posedge SDN, 0, 0, notifier);
       $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: TFCSNx , Mon Nov 24 11:09:05 CST 1997
`timescale 1ns / 10ps
`celldefine
module TFCSN2 (CP, CDN, SDN, Q, QN);

    input CP, CDN, SDN;
    output Q, QN;
    reg notifier;

    buf           (CDN_i, CDN);
    buf           (SDN_i, SDN);
    and     (CP_check, CDN_i, SDN_i);
    not     (D, Q_buf);
    tsmc_dff   (Q_buf, D, CP, CDN_i, SDN_i, notifier);
    buf          (Q, Q_buf);
    not     (QN_buf, Q_buf);
```

```
    and     (QN, QN_buf, SDN_i);

    reg flag;
    always @(CDN_i or SDN_i) begin
      if (!$test$plusargs("cdn_sdn_check_off")) begin
      if (flag == 1) begin
         if (CDN_i!==1'b0) begin
         $display("%m > CDN is released at time %.2fns.", $realtime);
         end
         if (SDN_i!==1'b0) begin
         $display("%m > SDN is released at time %.2fns.", $realtime);
         end
      end
      flag = ((CDN_i===1'b0)&&(SDN_i ===1'b0));
      if (flag == 1) begin
         $display("%m > Both CDN and SDN are enabled at time %.2fns.",
$realtime);
      end
    end
    end

    specify
       (CP => Q)=(0, 0);
       (CP => QN)=(0, 0);
       (CDN => Q)=(0, 0);
       (CDN => QN)=(0, 0);
       (SDN => Q)=(0, 0);
       (SDN => QN)=(0, 0);
       $recovery(posedge CDN,  posedge CP, 0, notifier);
       $hold(posedge CP , posedge CDN, 0, notifier);
       $recovery(posedge SDN,  posedge CP, 0, notifier);
```

```
        $hold(posedge CP , posedge SDN, 0, notifier);
        $width(posedge CP &&& CP_check, 0, 0, notifier);
        $width(negedge CP &&& CP_check, 0, 0, notifier);
        $width(posedge CDN, 0, 0, notifier);
        $width(negedge CDN, 0, 0, notifier);
        $width(posedge SDN, 0, 0, notifier);
        $width(negedge SDN, 0, 0, notifier);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XNR2Dx , Mon Nov 24 11:08:53 CST 1997
`timescale 1ns / 10ps
`celldefine
module XNR2D1 (A1, A2, ZN);
    input A1, A2;
    output ZN;
    xnor       (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XNR2Dx , Mon Nov 24 11:08:53 CST 1997
`timescale 1ns / 10ps
`celldefine
module XNR2D2 (A1, A2, ZN);
```

```
    input A1, A2;
    output ZN;
    xnor       (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XNR2Dx , Mon Nov 24 11:08:53 CST 1997
`timescale 1ns / 10ps
`celldefine
module XNR2D3 (A1, A2, ZN);
    input A1, A2;
    output ZN;
    xnor       (ZN, A1, A2);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XNR3Dx , Mon Nov 24 11:08:53 CST 1997
`timescale 1ns / 10ps
`celldefine
module XNR3D1 (A1, A2, A3, ZN);
    input A1, A2, A3;
```

```
    output ZN;
    xnor       (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XNR3Dx , Mon Nov 24 11:08:53 CST 1997
`timescale 1ns / 10ps
`celldefine
module XNR3D2 (A1, A2, A3, ZN);
    input A1, A2, A3;
    output ZN;
    xnor       (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XNR3Dx , Mon Nov 24 11:08:53 CST 1997
`timescale 1ns / 10ps
`celldefine
module XNR3D3 (A1, A2, A3, ZN);
```

```
    input A1, A2, A3;
    output ZN;
    xnor       (ZN, A1, A2, A3);
    specify
       (A1 => ZN)=(0, 0);
       (A2 => ZN)=(0, 0);
       (A3 => ZN)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XOR2Dx , Mon Nov 24 11:08:54 CST 1997
`timescale 1ns / 10ps
`celldefine
module XOR2D1 (A1, A2, Z);
    input A1, A2;
    output Z;
    xor    (Z, A1, A2);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XOR2Dx , Mon Nov 24 11:08:54 CST 1997
`timescale 1ns / 10ps
`celldefine
module XOR2D2 (A1, A2, Z);
```

```
    input A1, A2;
    output Z;
    xor    (Z, A1, A2);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XOR2Dx , Mon Nov 24 11:08:54 CST 1997
`timescale 1ns / 10ps
`celldefine
module XOR2D3 (A1, A2, Z);
    input A1, A2;
    output Z;
    xor    (Z, A1, A2);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XOR3Dx , Mon Nov 24 11:08:54 CST 1997
`timescale 1ns / 10ps
`celldefine
module XOR3D1 (A1, A2, A3, Z);
    input A1, A2, A3;
```

```
    output Z;
    xor    (Z, A1, A2, A3);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XOR3Dx , Mon Nov 24 11:08:54 CST 1997
`timescale 1ns / 10ps
`celldefine
module XOR3D2 (A1, A2, A3, Z);
    input A1, A2, A3;
    output Z;
    xor    (Z, A1, A2, A3);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// TSMC Standard Cell Function Library Ver
// Model Type: XOR3Dx , Mon Nov 24 11:08:54 CST 1997
`timescale 1ns / 10ps
`celldefine
module XOR3D3 (A1, A2, A3, Z);
```

```
    input A1, A2, A3;
    output Z;
    xor    (Z, A1, A2, A3);
    specify
       (A1 => Z)=(0, 0);
       (A2 => Z)=(0, 0);
       (A3 => Z)=(0, 0);
    endspecify
endmodule
`endcelldefine

// User-Defined Primitive        Edited by Yi-an Liang of ASIC/TSMC,
05/22/1995.
//                                          Modified by Hung-Chun Li
04/30/1997.
//*********************************************************************
*******
// This is a positive edge triggered User-Defined D flip-flop, with an
// active-low clear and an active-low set, CDN and SDN. Q will be cleared
to
// zero when the clear and set are simultaneously active, CDN = SDN = 0.
//
// The notifier is the user-defined response to timing check violations. it
// propogates an x value at the output, Q, when timing check violations occur.
//
// Note:
//    ?: denotes iteration of the table entry over the values 0, 1, and x.
// (vw): denotes value change from v to w, v and w can be any of 0, 1, x, ?
//    -: denotes no change.
//    *: denotes all the transitions.
```

```
primitive tsmc_dff (q, d, cp, cdn, sdn, notifier);
`protect
   output q;
   input d, cp, cdn, sdn, notifier;
   reg q;

   table
      ?   ?   0   ?   ? : ? : 0 ; // CDN dominate SDN
      ?   ?   1   0   ? : ? : 1 ; // SDN is set
      ?   ?   1   x   ? : 0 : x ; // SDN affect Q (added by mcchen 97/8/1)

      0 (01)  ?   1   ? : ? : 0 ; // Latch 0
      0   *   ?   1   ? : 0 : 0 ; // Keep 0 (D==Q)

      1 (01)  1   ?   ? : ? : 1 ; // Latch 1
      1   *   1   ?   ? : 1 : 1 ; // Keep 1 (D==Q)

      ? (1?)  1   1   ? : ? : - ; // ignore negative edge of clock
      ? (?0)  1   1   ? : ? : - ; // ignore negative edge of clock
      ?   ? (?1)  ?   ? : ? : - ; // ignore positive edge of CDN
      ?   ?   ? (?1)  ? : ? : - ; // ignore posative edge of SDN
      *   ?   ?   ?   ? : ? : - ; // ignore data change on steady clock

      ?   ?   ?   ?   * : ? : x ; // timing check violation
   endtable
`endprotect

endprimitive
primitive tsmc_dla (q, d, e, cdn, sdn, notifier);
`protect
   output q;
```

```
   input d, e, cdn, sdn, notifier;
   reg q;

   table

// d  e cdn sdn  noti : qt : qt+1
//
   1  1   1   1   ?   : ?  :  1  ; // Latch 1
   0  1   1   1   ?   : ?  :  0  ; // Latch 0
   0 (10) 1   1   ?   : ?  :  0  ; // Latch 0 after falling edge
   1 (10) 1   1   ?   : ?  :  1  ; // Latch 1 after falling edge

   *  0   ?   ?   ?   : ?  :  -  ; // no changes

   ?  ?   ?   0   ?   : ?  :  1  ; // preset to 1
   ?  0   1   *   ?   : 1  :  1  ;
   1  ?   1   *   ?   : 1  :  1  ;
   1  *   1   ?   ?   : 1  :  1  ;

   ?  ?   0   1   ?   : ?  :  0  ; // reset to 0
   ?  0   *   1   ?   : 0  :  0  ;
   0  ?   *   1   ?   : 0  :  0  ;
   0  *   ?   1   ?   : 0  :  0  ;

   ?  ?   ?   ?   *   : ?  :  x  ; // toggle notifier

   endtable
`endprotect

endprimitive
// User-Defined Primitive        Edited by Hung-Chun Li of ASIC/TSMC,
```

```
10/16/1996.
//*****************************************************************
*******
//
// Note:
//    ?: denotes iteration of the table entry over the values 0, 1, and x.

primitive tsmc_mux (q, d0, d1, s);
   output q;
   input s, d0, d1;

`protect
   table
   // d0  d1  s   : q
      0   ?   0   : 0 ;
      1   ?   0   : 1 ;
      ?   0   1   : 0 ;
      ?   1   1   : 1 ;
      0   0   x   : 0 ;
      1   1   x   : 1 ;
   endtable
`endprotect

endprimitive
// User-Defined Primitive        Edited by Yi-an Liang of ASIC/TSMC,
11/03/1994.
//                                            Modified by Hung-Chun Li
04/14/1997
//*****************************************************************
********
// This is a User-Defined unbuffered SR(2 cross-coupled NANDs) latch, with
```

```
an
// active-high set and active-high reset. Latch mode(SN=RN=1) may be entered
// only when complement output states have been established. SN=RN=0
generates
// a HIGH output on Q.
//
// The notifier is the user-defined response to timing check violations. it
// propogates an x value at the output, Q, when timing check violations occur.
//
// Note:
//    ?: denotes iteration of the table entry over the values 0, 1, and x.
//    -: denotes no change.
//    *: denotes all the transitions.


primitive tsmc_ndla (q, rn, sn, notifier);
`protect
   output q;
   input rn, sn, notifier;
   reg q;

   table
      ?   0   ?  : ? : 1 ; // Set dominate Clear
      0   1   ?  : ? : 0 ; // Clear
//      1   ?   ?  : 1 : 1 ; // Hold 1 or Set
//      ?   1   ?  : 0 : 0 ; // Hold 0 or Clear
//      1   1   ?  : ? : - ; // Hold original state
      1   p   ?  : ? : - ; // Hold original state
      p   1   ?  : ? : - ; // Hold original state
      ?   ?   *  : ? : x ; // Timing check violation
   endtable
```

```
`endprotect

endprimitive
// User-Defined Primitive        Edited by Yi-an Liang of ASIC/TSMC,
11/03/1994.
//                                           Modified by Hung-Chun Li
04/14/1997
//*****************************************************************
********
// This is a User-Defined SR(2 cross-coupled NORs) latch, with an active-high
// set and active-high reset. Latch mode(S=R=0) may be entered only when
// complement output states have been established. S=R=1 generates a LOW
// output on Q.
//
// The notifier is the user-defined response to timing check violations. it
// propogates an x value at the output, Q, when timing check violations occur.
//
// Note:
//    ?: denotes iteration of the table entry over the values 0, 1, and x.
//    -: denotes no change.
//    *: denotes all the transitions.

primitive tsmc_nrla (q, r, s, notifier);
`protect
   output q;
   input r, s, notifier;
   reg q;

   table
      1   ?   ?  : ? : 0 ; // Reset dominate Set
      0   1   ?  : ? : 1 ; // Set
```

```
//       ?   0   ?  : 0 : 0 ; // Reset or Hold 0
//       0   ?   ?  : 1 : 1 ; // Set or Hold 1
//       0   0   ?  : ? : - ; // Hold original state
      0   n   ?  : ? : - ; // Hold original state
      n   0   ?  : ? : - ; // Hold original state
      ?   ?   *  : ? : x ; // Timing Check violation
   endtable
`endprotect

endprimitive
```

附录 3 Dracula 命令文件

1． DRC 命令文件

```
;*********************************************************************
*
;
;DRC CHECKS FOR 0.5 MICRON TRIBLE METAL SINGLE POLY CMOS PROCESS OF TSMC;
;                            Version  1.0
;                  by DENG HAIFEI    MARCH 23, 1999
;
;*********************************************************************
*
*DESCRIPTION
  INDISK      = /EDAHOME01/students/dhf/sram/drc/mux2.gds
  OUTDISK     = /EDAHOME01/students/dhf/sram/drc/drc.gds
  WORK-DIR    = /EDAHOME01/students/dhf/sram/drc/
  PRIMARY     = MUX2
  SYSTEM      = GDS2
  SCALE       = 0.001 MICRON
  RESOLUTION  = 0.1 MICRON
  PRINTFILE   = drcprt
  LISTERROR   = YES
  PREINQUERY  = YES
  KEEPDATA    = YES
  CNAMES-CSEN  = YES
 *END
;
;*********************************************************
;
```

```
;                    INPUT-LAYER BLOCK
;
;**********************************************************
*INPUT-LAYER
  NDIFF = 1
  PDIFF = 2
  DIFF  = 3
  PWELL = 6
  VIA2  = 7
  VIA   = 8
  NWELL = 12
  PAD   = 29
  POLY  = 35
  ESD   = 44
  METAL1 = 45  TEXT = 61
  METAL2 = 50
  METAL3 = 46
  CONT  = 55
  TEXT  = 61

  SUBSTRATE = BULK 60; Layer created for reverse mask
;
  CONNECT-LAY = NWELL PWELL NSD PSD POLY METAL1 METAL2 METAL3
;
; *** The conductor layer of the CMOS process ***
;
*END
;
;**********************************************************
;
;                     OPERATION BLOCK
```

```
;      CREAT THE DEVICE LAYERS AND CONDUCTOR LAYERS
;
;*********************************************************
*OPERATION

  AND   DIFF    PDIFF   PREGION
  NOT   DIFF    PREGION NREGION
  AND   POLY     NREGION    NGATE
  AND   POLY    PREGION PGATE
  OR    PGATE   NGATE   GATE
  NOT   NREGION NGATE   NSD
  NOT   PREGION PGATE   PSD
  OR    NSD     PSD SRCDRN
  NOT   BULK    PWELL   NWSUB
  NOT   NWSUB   NWELL   SUB
  AND   DIFF    NWELL   DIFNW
  AND   DIFF    PWELL   DIFPW
  OR    DIFNW   DIFPW   DIFWE
  AND   PWELL   PSD PSDPW
  AND   NWELL   NSD NSDNW

;*********************************************************
;
;            CONNECT OPERATION FOR CMOS PROCESS
;
;*********************************************************
;
CONNECT METAL1 POLY BY CONT
CONNECT METAL1 NSD BY CONT
CONNECT METAL1 PSD BY CONT
CONNECT PSD PWELL BY PSDPW
```

```
CONNECT NSD NWELL BY NSDNW
CONNECT METAL2 METAL1 BY VIA
CONNECT METAL3 METAL2  BY VIA2
;
;                          DRC CHECKS
;

;*********************************************************************
;                     NWELL RULES
;*********************************************************************
 WIDTH      NWELL   LT  2.5     OUTPUT  DRC01 31
 EXT        NWELL       LT  4.0 OUTPUT  DRC01 32
;
;**********************************************************************
;       THIN  OXIDE   RULE
;**********************************************************************
  SELECT     DIFWE  CUT POLY    DFWP
  WIDTH     DFWP    LT  0.6 OUTPUT  DRC02 41
  NOT       DIFF    DFWP    IDFWP
  WIDTH     IDFWP   LT  0.5 OUTPUT  DRC02 42
  SELECT     DIFF   INSIDE  NWELL   DFINNW
  SELECT     DIFF   OUTSIDE NWELL   DFOUNW
  OR        DFINNW DFOUNW    DFIONW
  EXT        DFIONW LT  0.9 OUTPUT  DRC02 43
  AND       DFINNW NDIFF   NDFINNW
  ENC[CP]    NWELL  NDFINNW LT 0.3 OUTPUT  DRC02 44
  NOT       NREGIONNDFINNW NDFOUNW
  EXT[C]     NWELL  NDFOUNW LT 3.3 OUTPUT  DRC02 45
  SELECT     PREGION    INSIDE  NWELL   PDFINNW
  ENC[CP]    NWELL  PDFINNW LT 1.5 OUTPUT  DRC02 46
  NOT       PREGIONPDFINNW PDFOUNW
```

```
  EXT[C]     NWELL  PDFOUNW LT 0.3     OUTPUT  DRC02 47
  SELECT     POLY   CUT PREGION POPR
  EXT[C]     POPR   NREGION LT  0.6 OUTPUT  DRC02 48
  SELECT     POLY   CUT NREGION PONR
  EXT[C]     PONR   PREGION LT  0.6 OUTPUT  DRC02 50
  SELECT     PREGION    TOUCH   NREGION PRTNR
  NOT        PREGIONPRTNR   PRNTNR
  EXT[C]     PRNTNR NREGION LT  0.9 OUTPUT  DRC02 49
;****************************************************************
;              POLY    RULE
;****************************************************************
  WIDTH    PGATE    LT  0.5 OUTPUT  DRC03   51
  WIDTH    NGATE    LT  0.5 OUTPUT  DRC03   52
  OR       POPR     PONR    POTR
  NOT      POLY     POTR    IPOLY
  WIDTH    IPOLY    LT  0.5 OUTPUT  DRC03   53
  EXT   POLY        LT  0.6 OUTPUT  DRC03   54
  NOT      BULK     DIFF    OXFID
  SIZE     GATE     BY  0.26    OGATE
  NOT      POLY     OGATE   NOGPOLY
  AND      NOGPOLY      OXFID   POOXFI
  EXT[C]   POOXFI  DIFF    LT  0.25    OUTPUT  DRC03   55
  ENC[CP]  DIFF        POLY    LT  0.65    OUTPUT  DRC03   56
  WIDTH    POOXFI  LT  0.5 OUTPUT  DRC03   57
;********************************************************************
;      P+ S/D RULE
;*********************************************************************
  WIDTH    PDIFF    LT  0.75    OUTPUT  DRC04   41
  EXT       PDIFF   LT  0.75    OUTPUT  DRC04   42
  EXT[C]   PDIFF    DIFF    LT  0.6 OUTPUT  DRC04   43
  ENC[CP]  PDIFF    PGATE   LT  0.6 OUTPUT  DRC04   44
```

```
  ENC[CP]  PDIFF  NGATE  LT  0.6 OUTPUT  DRC04  48
  INT[C]   PDIFF  DIFF   LT  0.6 OUTPUT  DRC04  45
  ENC[CP]  PDIFF  PREGION LT 0.3 OUTPUT  DRC04  46
  EXT[C]   PDIFF  NDIFF  LT  0.3 OUTPUT  DRC04  47
;*********************************************************************
;       N+ S/D RULE
;**********************************************************************
  WIDTH   NDIFF   LT  0.75   OUTPUT  DRC05  41
  EXT      NDIFF  LT  0.75   OUTPUT  DRC05  42
  EXT[C]   NDIFF  DIFF   LT  0.6 OUTPUT  DRC05  43
  ENC[CP]  NDIFF  PGATE  LT  0.6 OUTPUT  DRC05  44
  ENC[CP]  NDIFF  NGATE  LT  0.6 OUTPUT  DRC05  48
  INT[C]   NDIFF  DIFF   LT  0.6 OUTPUT  DRC05  45
  ENC[CP]  NDIFF  NREGION LT 0.3 OUTPUT  DRC05  46
  EXT[C]   NDIFF  PDIFF  LT  0.3 OUTPUT  DRC05  47
;***********************************************************************
;   ESD IMPLANTATION RULE
;***********************************************************************
;IT WILL BE WRITTEN LATER.
;***********************************************************************
;   CONTACT RULE
;***********************************************************************
;  WIDTH      CONT  LT  0.5 OUTPUT  DRC07  71
;  WIDTH      CONT  GT  0.5 OUTPUT  DRC07  70
  EXT      CONT LT  0.5 OUTPUT  DRC07  72
  AND     CONT  DIFF   CONDIF
  EXT[C]  CONDIF GATE  LT  0.4 OUTPUT  DRC07 73
  AND     CONT  POLY   CONPO
  EXT[C]  CONPO DIFF   LT  0.5 OUTPUT  DRC07 74
  ENC[C]  DIFF CONT    LT  0.25   OUTPUT  DRC07 75
  SELECT  CONT INSIDE  DIFF   CONINDI
```

```
  SIZE    CONINDI  BY  0.3 CONIDO
  SELECT  CONIDO  CUT DIFF CONCUDI   OUTPUT  DRC07 76
  SELECT  CONT INSIDE POLY   CONINPO
  ENC[CP] POLY CONTINPO   LT  0.3 OUTPUT  DRC07 77
  ENC[CP] PDIFF    CONT    LT  0.3 OUTPUT  DRC07 78
  ENC[CP] NDIFF    CONT    LT  0.3 OUTPUT  DRC07 79
  SELECT  CONT CUT GATE    CONGA  OUTPUT  DRC07 80

;**************************************************************************
;       METAL-1 RULE
;**************************************************************************
  WIDTH   METAL1   LT  0.6 OUTPUT  DRC08   81
  EXT      METAL1   LT  0.6 OUTPUT  DRC08   82
  ENC[CP] METAL1   CONT    LT  0.25    OUTPUT  DRC08   83
  WIDTH   METAL1   SELGT   10  M110
  EXT[C]  METAL1   M110    LT  1.0 OUTPUT  DRC08   84
;***************************************************************************
;   VIA RULE
;**************************************************************************
;  WIDTH      VIA   LT  0.6 OUTPUT  DRC09   91
;  WIDTH      VIA   GT  0.6 OUTPUT  DRC09   98
  EXT     VIA   LT  0.6 OUTPUT  DRC09   92
  ENC[C]  METAL1   VIA LT  0.3 OUTPUT  DRC09 95
  ENC[C]  M110 VIA LT  0.5 OUTPUT  DRC09 96
  AND   VIA CONT    VICON
  AND   VICON  VIA2    VIA12CO OUTPUT DRC09   97
  EXT[CP]  VIA2    VICON   LT  0.5 OUTPUT  DRC09 99

;***************************************************************************
;   METAL2  RULE
;***************************************************************************
```

```
  WIDTH   METAL2   LT  0.7 OUTPUT  DRC10   81
  EXT      METAL2   LT  0.7 OUTPUT  DRC10   82
  ENC[CP] METAL2   VIA LT  0.3 OUTPUT  DRC10   83
  WIDTH   METAL2   SELGT   10  M210
  ENC[CP] M210 VIA LT  0.5 OUTPUT  DRC10   85
  EXT[C]  METAL2   M110    LT  1.0 OUTPUT  DRC10   84
;*********************************************************************
;   VIA2    RULE
;**********************************************************************
;  WIDTH      VIA2  LT  0.6 OUTPUT  DRC11   91
;  WIDTH      VIA2  GT  0.6 OUTPUT  DRC11   97
  EXT   VIA2   LT  0.6 OUTPUT  DRC11   92
  ENC[C]  METAL2   VIA2    LT  0.3 OUTPUT  DRC11 95
  ENC[C]  M210 VIA2    LT  0.5 OUTPUT  DRC11 96
  AND   VIA2    VIA VIA12
  EXT[CP]  VIA12   CONT    LT  0.5 OUTPUT  DRC11 98

;**********************************************************************
;   METAL3  RULE
;**********************************************************************
  WIDTH   METAL3   LT  0.8 OUTPUT  DRC12   81
  EXT  METAL3  LT  0.7 OUTPUT  DRC12   82
  ENC[CP] METAL3   VIA2    LT  0.3 OUTPUT  DRC12   83
  WIDTH   METAL3   SELGT   10  M310
  ENC[CP] M310 VIA2    LT  0.5 OUTPUT  DRC12   85
  EXT[C]  METAL3   M310    LT  1.0 OUTPUT  DRC12   84
;
*END
```

2． ERC 命令文件

```
;*********************************************************************
*
;
;DRC CHECKS FOR 0.5 MICRON TRIBLE METAL SINGLE POLY CMOS PROCESS OF TSMC;
;                             Version  1.0
;                   by DENG HAIFEI    MARCH 23, 1999
;
;*********************************************************************
*
*DESCRIPTION
  INDISK      = /EDAHOME01/students/dhf/sram/drc/mux2.gds
  OUTDISK     = /EDAHOME01/students/dhf/sram/drc/drc.gds
  WORK-DIR    = /EDAHOME01/students/dhf/sram/drc/
  PRIMARY     = MUX2
  SYSTEM      = GDS2
  SCALE       = 0.001 MICRON
  RESOLUTION  = 0.1 MICRON
  PRINTFILE   = ercprt
  LISTERROR   = YES
  PREINQUERY  = YES
  KEEPDATA    = YES
  CNAMES-CSEN  = YES
 *END
;
;**********************************************************
;
;                    INPUT-LAYER BLOCK
;
;**********************************************************
*INPUT-LAYER
  NDIFF = 1
```

```
  PDIFF = 2
  DIFF  = 3
  PWELL = 6
  VIA2  = 7
  VIA   = 8
  NWELL = 12
  PAD   = 29
  POLY  = 35
  ESD   = 44
  METAL1 = 45  TEXT = 61
  METAL2 = 50
  METAL3 = 46
  CONT  = 55
  TEXT  = 61

  SUBSTRATE = BULK 60; Layer created for reverse mask
;
  CONNECT-LAY = NWELL PWELL NSD PSD POLY METAL1 METAL2 METAL3
;
; *** The conductor layer of the CMOS process ***
;
*END
;
;**********************************************************
;
;                     OPERATION BLOCK
;     CREAT THE DEVICE LAYERS AND CONDUCTOR LAYERS
;
;**********************************************************
*OPERATION
```

```
  AND   DIFF    PDIFF   PREGION
  NOT   DIFF    PREGION NREGION
  AND   POLY     NREGION    NGATE
  AND   POLY    PREGION PGATE
  OR    PGATE   NGATE   GATE
  NOT   NREGION NGATE   NSD
  NOT   PREGION PGATE   PSD
  OR    NSD     PSD SRCDRN
  NOT   BULK    PWELL   NWSUB
  NOT   NWSUB   NWELL   SUB
  AND   DIFF    NWELL   DIFNW
  AND   DIFF    PWELL   DIFPW
  OR    DIFNW   DIFPW   DIFWE
  AND   PWELL   PSD PSDPW
  AND   NWELL   NSD NSDNW

;**********************************************************
;
;             CONNECT OPERATION FOR CMOS PROCESS
;
;**********************************************************
;
CONNECT METAL1 POLY BY CONT
CONNECT METAL1 NSD BY CONT
CONNECT METAL1 PSD BY CONT
CONNECT PSD PWELL BY PSDPW
CONNECT NSD NWELL BY NSDNW
CONNECT METAL2 METAL1 BY VIA
CONNECT METAL3 METAL2  BY VIA2
;
```

```
;************************************************************************
**
;************************************************************************
**
;
;       DEFINE ELEMENTS OF CMOS PROCESS
;
;************************************************************************
**
;************************************************************************
**
  ELEMENT  MOS[N] NGATE  POLY   NSD PWELL
  ELEMENT  MOS[P] PGATE  POLY   PSD NWELL
;************************************************************************
**
;************************************************************************
**
;
;       ERC CHECKS
;
;************************************************************************
**
;************************************************************************
**
;
   MULTILABOUTPUT SHORTS 40
   SAMELAB OUTPUT OPENS  40
   NDCOUNT MOS[N] NSD GT  2   OUTPUT DEVERR1 45
   NDCOUNT MOS[P] PSD GT  2   OUTPUT DEVERR2 45
   ELCOUNT MOS ALL EQ  0   OUTPUT FLOAT  40
   ECONNECTMOS[N] NSD CONN   VDD &
```

```
   ECONNECTMOS[N]  NSD CONN    VSS OUTPUT  VDVSN   48
   ECONNECTMOS[P]  PSD CONN    VDD &
   ECONNECTMOS[P]  PSD CONN    VSS OUTPUT  VDVSP   48
   ECONNECTMOS[P]  POLY    CONN    VDD OUTPUT  GATVDD  49
   ECONNECTMOS[N]  POLY    CONN    VSS OUTPUT  GATVSS  49
   ECONNECTMOS[P]  PSD CONN    VSS OUTPUT  PSDVSS  50
   ECONNECTMOS[N]  NSD CONN    VDD OUTPUT  NSDVDD  50
;PATHCHK   LEVEL   1   OUTPUT  NOVDD   40
;  PATHCHK LEVEL   2   OUTPUT  NOVSS   40
;  PATHCHK LEVEL   3   OUTPUT  NOPWGR  40
;  PATHCHK LEVEL   4   OUTPUT  NOALL   40
   ELCOUNT MOS ALL EQ  1   OUTPUT  ONEDEV  40
   NDCOUNT MOS[N]  NSD EQ  2   &
   NDCOUNT MOS[N]  ALL EQ  2   OUTPUT  NDEPL  55
   NDCOUNT MOS ALL EQ  1   OUTPUT  MIXUP   56
   LCONNECTPWELL   DISC    GND OUTPUT  FLOWEL  57
   LCONNECTNWELL   DISC    VDD OUTPUT  FLOWEN  57

;
;
*END
```

3. LVS 命令文件

```
;**************************************************************************
*
;
;DRC CHECKS FOR 0.5 MICRON TRIBLE METAL SINGLE POLY CMOS PROCESS OF TSMC;
;                              Version  1.0
;                    by DENG HAIFEI    MARCH 23, 1999
;
```

```
;**********************************************************************
*
*DESCRIPTION
  INDISK      = /EDAHOME01/students/dhf/sram/drc/mux2.gds
  OUTDISK     = /EDAHOME01/students/dhf/sram/drc/drc.gds
  WORK-DIR    = /EDAHOME01/students/dhf/sram/drc/
  PRIMARY     = MUX2
  MODE       = EXEC NOW
  SYSTEM      = GDS2
  SCALE       = 0.001 MICRON
  RESOLUTION  = 0.1 MICRON
  PRINTFILE   = ercprt
  LISTERROR   = YES
  PREINQUERY  = YES
  KEEPDATA    = YES
  CNAMES-CSEN  = YES
  SCHEMATIC     =LVSLOGIC
 *END
;
;*********************************************************
;
;                    INPUT-LAYER BLOCK
;
;*********************************************************
*INPUT-LAYER
  NDIFF = 1
  PDIFF = 2
  DIFF  = 3
  PWELL = 6
  VIA2  = 7
  VIA   = 8
```

```
  NWELL = 12
  PAD   = 29
  POLY  = 35
  ESD   = 44
  METAL1 = 45  TEXT = 61
  METAL2 = 50
  METAL3 = 46
  CONT  = 55
  TEXT  = 61

  SUBSTRATE = BULK 60; Layer created for reverse mask
;
  CONNECT-LAY = NWELL PWELL NSD PSD POLY METAL1 METAL2 METAL3
;
; *** The conductor layer of the CMOS process ***
;
*END
;
;***********************************************************
;
;                     OPERATION BLOCK
;      CREAT THE DEVICE LAYERS AND CONDUCTOR LAYERS
;
;***********************************************************
*OPERATION

  AND   DIFF    PDIFF   PREGION
  NOT   DIFF    PREGION NREGION
  AND   POLY     NREGION    NGATE
  AND   POLY    PREGION PGATE
  OR    PGATE   NGATE   GATE
```

```
  NOT   NREGION NGATE   NSD
  NOT   PREGION PGATE   PSD
  OR    NSD     PSD SRCDRN
  NOT   BULK    PWELL   NWSUB
  NOT   NWSUB   NWELL   SUB
  AND   DIFF    NWELL   DIFNW
  AND   DIFF    PWELL   DIFPW
  OR    DIFNW   DIFPW   DIFWE
  AND   PWELL   PSD PSDPW
  AND   NWELL   NSD NSDNW

;**********************************************************
;
;            CONNECT OPERATION FOR CMOS PROCESS
;
;**********************************************************
;
CONNECT METAL1 POLY BY CONT
CONNECT METAL1 NSD BY CONT
CONNECT METAL1 PSD BY CONT
CONNECT PSD PWELL BY PSDPW
CONNECT NSD NWELL BY NSDNW
CONNECT METAL2 METAL1 BY VIA
CONNECT METAL3 METAL2  BY VIA2
;


;
;       DEFINE ELEMENTS OF CMOS PROCESS
;
;***********************************************************************
**
```

```
;***********************************************************************
**
  ELEMENT  MOS[N]  NGATE   POLY    NSD PWELL
  ELEMENT  MOS[P]  PGATE   POLY    PSD NWELL
;***********************************************************************
**
;***********************************************************************
**
;
;       ERC CHECKS
;
;***********************************************************************
**
;***********************************************************************
**
;
   MULTILABOUTPUT  SHORTS  40
   SAMELAB OUTPUT  OPENS   40
   NDCOUNT MOS[N]  NSD GT  2   OUTPUT  DEVERR1 45
   NDCOUNT MOS[P]  PSD GT  2   OUTPUT  DEVERR2 45
   ELCOUNT MOS ALL EQ  0   OUTPUT  FLOAT   40
   ECONNECTMOS[N]  NSD CONN    VDD &
   ECONNECTMOS[N]  NSD CONN    VSS OUTPUT  VDVSN   48
   ECONNECTMOS[P]  PSD CONN    VDD &
   ECONNECTMOS[P]  PSD CONN    VSS OUTPUT  VDVSP   48
   ECONNECTMOS[P]  POLY    CONN    VDD OUTPUT  GATVDD  49
   ECONNECTMOS[N]  POLY    CONN    VSS OUTPUT  GATVSS  49
   ECONNECTMOS[P]  PSD CONN    VSS OUTPUT  PSDVSS  50
   ECONNECTMOS[N]  NSD CONN    VDD OUTPUT  NSDVDD  50
   PATHCHK LEVEL   1   OUTPUT  NOVDD   40
   PATHCHK LEVEL   2   OUTPUT  NOVSS   40
```

```
   PATHCHK LEVEL   3   OUTPUT  NOPWGR  40
   PATHCHK LEVEL   4   OUTPUT  NOALL   40
   ELCOUNT MOS ALL EQ  1   OUTPUT  ONEDEV  40
   NDCOUNT MOS[N]  NSD EQ  2   &
   NDCOUNT MOS[N]  ALL EQ  2   OUTPUT  NDEPL  55
   NDCOUNT MOS ALL EQ  1   OUTPUT  MIXUP   56
   LCONNECTPWELL   DISC    VSS OUTPUT  FLOWEL  57
   LCONNECTNWELL   DISC    VDD OUTPUT  FLOWEN  57
   LVSCHK
;
;
*END
```

4. LPE 命令文件

```
;************************************************************************
*
;
;DRC CHECKS FOR 0.5 MICRON TRIBLE METAL SINGLE POLY CMOS PROCESS OF TSMC;
;                              Version  1.0
;                    by DENG HAIFEI    MARCH 23, 1999
;
;************************************************************************
*
*DESCRIPTION
  INDISK      = /EDAHOME01/students/dhf/sram/drc/mux2.gds
  OUTDISK     = /EDAHOME01/students/dhf/sram/drc/lpe.gds
  WORK-DIR    = /EDAHOME01/students/dhf/sram/drc/
  PRIMARY     = MUX2
  MODE        = EXEC NOW
  SYSTEM      = GDS2
```

```
  SCALE        = 0.001 MICRON
  RESOLUTION  = 0.1 MICRON
  PRINTFILE   = ercprt
  DIODESEQ    = A1 P1
  MODEL       = MOS[N],N MOS[P],P DIO[N],N DIO[P],P
  UNIT       = CAPACITANCE,PF
  LISTERROR   = YES
  PREINQUERY  = YES
  KEEPDATA    = YES
  CNAMES-CSEN  = YES
  SCHEMATIC     =LVSLOGIC
 *END
;
;**********************************************************
;
;                   INPUT-LAYER BLOCK
;
;**********************************************************
*INPUT-LAYER
  NDIFF = 1
  PDIFF = 2
  DIFF  = 3
  PWELL = 6
  VIA2  = 7
  VIA   = 8
  NWELL = 12
  PAD   = 29
  POLY  = 35
  ESD   = 44
  METAL1 = 45  TEXT = 61
  METAL2 = 50
```

```
  METAL3 = 46
  CONT  = 55
  TEXT  = 61

  SUBSTRATE = BULK 60; Layer created for reverse mask
;
  CONNECT-LAY = NWELL PWELL NSD PSD POLY METAL1 METAL2 METAL3
;
; *** The conductor layer of the CMOS process ***
;
*END
;
;**********************************************************
;
;                    OPERATION BLOCK
;     CREAT THE DEVICE LAYERS AND CONDUCTOR LAYERS
;
;**********************************************************
*OPERATION

  AND   DIFF    PDIFF   PREGION
  NOT   DIFF    PREGION NREGION
  AND   POLY     NREGION    NGATE
  AND   POLY    PREGION PGATE
  OR    PGATE   NGATE   GATE
  NOT   NREGION NGATE   NSD
  NOT   PREGION PGATE   PSD
  OR    NSD     PSD SRCDRN
  NOT   BULK    PWELL   NWSUB
  NOT   NWSUB   NWELL   SUB
  AND   DIFF    NWELL   DIFNW
```

```
  AND   DIFF    PWELL   DIFPW
  OR    DIFNW   DIFPW   DIFWE
  AND   PWELL   PSD PSDPW
  AND   NWELL   NSD NSDNW

;**********************************************************
;
;            CONNECT OPERATION FOR CMOS PROCESS
;
;**********************************************************
;
CONNECT METAL1 POLY BY CONT
CONNECT METAL1 NSD BY CONT
CONNECT METAL1 PSD BY CONT
CONNECT PSD PWELL BY PSDPW
CONNECT NSD NWELL BY NSDNW
CONNECT METAL2 METAL1 BY VIA
CONNECT METAL3 METAL2  BY VIA2
;***********************************************************************
**
;***********************************************************************
**
;
;       DEFINE ELEMENTS OF CMOS PROCESS
;
;***********************************************************************
**
;***********************************************************************
**
 ELEMENT   MOS[N]  NGATE   POLY    NSD PWELL
  ELEMENT   MOS[P]  PGATE   POLY    PSD NWELL
```

```
;
;*************************************************************************
*******
;*************************************************************************
*******
;       LPE CHECK
;*************************************************************************
*******
;*************************************************************************
*******
;
;*************************************************************************
*******
;   LOGIC OPERATION    DEFINING THE PARASITICCAPACITORS AND DIODES
;*************************************************************************
*******
;
  AND   METAL1  POLY    MPOLY   ; METAL1 OT POLY CAPACITORS
  NOT   METAL1  POLY    METNP   ; REMOVE POLY UNDERNEATH METAL1
  AND   METNP   NSD MNSD    ; METAL TO N+ SOURCE/DRAIN CAPACITORS
  AND   METNP   PSD MPSD    ; METAL TO P+ SOURCE/DRAIN CAPACITORS
  NOT   METNP   NSD M1
  NOT   M1  PSD M2
  AND   M2  PWELL   MPWELL  ; METAL TO PWELL CAPACITORS
  AND   M2  NWELL   MNWELL  ; METAL TO NWELL CAPACITORS
;
  NOT   POLY    GATE    P2
  AND   P2  PWELL   POWELL  ;POLY TO PWELL CAPACITORS
  AND   P2  NWELL   PONW    ; POLY TO NWELL CAPACITORS
;
  AND   METAL2  METAL1  METAL12 ; METAL2 TO METAL1 CAPACITORS
```

```
  NOT   METAL2  METAL1  MET2N1
  AND   MET2N1  POLY    M2POLY  ; METAL2 OT POLY CAPACITORS
  NOT   MET2N1  POLY    MET2NP  ; REMOVE POLY UNDERNEATH METAL2
  AND   MET2NP  NSD M2NSD   ; METAL2 TO N+ SOURCE/DRAIN CAPACITORS
  AND   MET2NP  PSD M2PSD   ; METAL2 TO P+ SOURCE/DRAIN CAPACITORS
  NOT   MET2NP  NSD M21
  NOT   M21 PSD M22
  AND   M22 PWELL   M2PWELL ; METAL2 TO PWELL CAPACITORS
  AND   M22 NWELL   M2NWELL ; METAL2 TO NWELL CAPACITORS
;
  AND   METAL3  METAL2  METAL23  ;METAL3 TO METAL2 C
  NOT   METAL3  METAL2  MET3N2
  AND   MET3N2  METAL1  METAL13 ; METAL3 TO METAL1 CAPACITORS
  NOT   MET3N2  METAL1  MET3N1
  AND   MET3N1  POLY    M3POLY  ; METAL3 OT POLY CAPACITORS
  NOT   MET3N1  POLY    MET3NP  ; REMOVE POLY UNDERNEATH METAL3
  AND   MET3NP  NSD M3NSD   ; METAL3 TO N+ SOURCE/DRAIN CAPACITORS
  AND   MET3NP  PSD M3PSD   ; METAL3 TO P+ SOURCE/DRAIN CAPACITORS
  NOT   MET3NP  NSD M31
  NOT   M31 PSD M32
  AND   M32 PWELL   M3PWELL ; METAL3 TO PWELL CAPACITORS
  AND   M32 NWELL   M3NWELL ; METAL3 TO NWELL CAPACITORS
;
  AND   NSD PWELL   NPDIO
  NOT   PSD PWELL   PPDIO

;*************************************************************************
*******
;   DEFINE TRANSITORS ELEMENTS AND PARASITIC ELEMENTS
;*************************************************************************
*******
```

```
; ELEMENT MOS[N]   NGATE   POLY    NSD PWELL
; ELEMENT MOS[P]   PGATE   POLY    PSD NWELL
;
  PARASITIC DIO[N] NPDIO   PWELL   NPDIO
  PARASITIC DIO[P] PPDIO   PPDIO   NWELL
;
  PARASITIC CAP[A] MPOLY   METAL1  POLY
  ATTRIBUTE CAP[A] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[B] MNSD    METAL1  NSD
  ATTRIBUTE CAP[B] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[C] MPSD    METAL1  PSD
  ATTRIBUTE CAP[C] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[D] MPWELL  METAL1  PWELL
  ATTRIBUTE CAP[D] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[E] MNWELL  METAL1  NWELL
  ATTRIBUTE CAP[E] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[F] POWELL  POLY    PWELL
  ATTRIBUTE CAP[F] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[G] PONW    POLY    NWELL
  ATTRIBUTE CAP[G] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[H] MPSD    METAL1  PSD
  ATTRIBUTE CAP[H] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
; PARASITIC CAP[I] METAL12 METAL2  METAL1
```

```
;  ATTRIBUTE CAP[I]    0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;

  PARASITIC CAP[J] M2POLY  METAL2  POLY
  ATTRIBUTE CAP[J] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[K] M2NSD   METAL2  NSD
  ATTRIBUTE CAP[K] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[L] M2PSD   METAL2  PSD
  ATTRIBUTE CAP[L] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[M] M2PWELL METAL2  PWELL
  ATTRIBUTE CAP[M] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[N] M2NWELL METAL2  NWELL
  ATTRIBUTE CAP[N] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
;  PARASITIC CAP[O]    METAL23 METAL3  METAL2
; ; ATTRIBUTE CAP[O]   0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
;  PARASITIC CAP[P]    METAL13 METAL3  METAL1
; ; ATTRIBUTE CAP[P]   0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;

  PARASITIC CAP[Q] M3POLY  METAL3  POLY
  ATTRIBUTE CAP[Q] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[R] M3NSD   METAL3  NSD
  ATTRIBUTE CAP[R] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
```

```
  PARASITIC CAP[S] M3PSD   METAL3  PSD
  ATTRIBUTE CAP[S] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[T] M3PWELL METAL3  PWELL
  ATTRIBUTE CAP[T] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  PARASITIC CAP[U] M3NWELL METAL3  NWELL
  ATTRIBUTE CAP[U] 0.00005  ; ATTRIBUTE (PF/SQ MICRON)
;
  LPECHK
;
  LPESELECT[S] MOS[P] &
  LPESELECT[S] MOS[N] &
  LPESELECT[S] DIO[P] &
  LPESELECT[S] DIO[N] &
  LPESELECT[S] CAP[A] GT 0.0 &
  LPESELECT[S] CAP[B] GT 0.0 &
  LPESELECT[S] CAP[C] GT 0.0 &
  LPESELECT[S] CAP[D] GT 0.0 &
  LPESELECT[S] CAP[E] GT 0.0 &
  LPESELECT[S] CAP[F] GT 0.0 &
  LPESELECT[S] CAP[G] GT 0.0 &
  LPESELECT[S] CAP[H] GT 0.0 &
;  LPESELECT[S] CAP[I] GT 0.0 &
  LPESELECT[S] CAP[J] GT 0.0 &
  LPESELECT[S] CAP[K] GT 0.0 &
  LPESELECT[S] CAP[L] GT 0.0 &
  LPESELECT[S] CAP[M] GT 0.0 &
  LPESELECT[S] CAP[N] GT 0.0 &
;  LPESELECT[S] CAP[O] GT 0.0 &
;  LPESELECT[S] CAP[P] GT 0.0 &
```

```
  LPESELECT[S] CAP[Q] GT 0.0 &
  LPESELECT[S] CAP[R] GT 0.0 &
  LPESELECT[S] CAP[S] GT 0.0 &
  LPESELECT[S] CAP[C] GT 0.0 &
  LPESELECT[S] CAP[U] GT 0.0   OUTPUT MYSPI
;

*END
```